ISSN (Online): 2455-7838
SJIF Impact Factor : 6.093

EPRA International Journal of

Research & Development (IJRD)

Monthly Peer Reviewed & Indexed
International Online Journal

Volume: 4, Issue:5, May 2019

Published By
EPRA Publishing

SJIF Impact Factor: 6.093 Volume: 4 | Issue: 5 | May| 2019 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Peer Reviewed Journal

WARAY SCRIPTING LANGUAGE (WSL): A MOTHER TONGUE - BASED SCRIPTING LANGUAGE

Rolando Real Codilan
Instructor, College of Computer Studies, Eastern Samar State University, Borongan City Philippines

ABSTRACT

The objective of this study was to develop a waray scripting language to introduce computer programming using mother tongue based scripting language to the beginners especially the first year college students enrolled in computer related programs in Eastern Samar State University and to evaluate the waray scripting language in terms of its usability, readability, and writabilty. This study used racket programming language an open-source programming language to develop WSL, racket helps programmers develop and quickly deploy new language. WSL was subjected to software evaluation in terms of usability, readability and writability was based on (ISO 9126) Software Product Quality Metrics. This study used developmental-evaluative research design. The result shows that WSL a mother-tongue based scripting language is of acceptable and is compliant to the ISO software standard with the variables obtained the average weighted mean rated as 3.5 Acceptable in general. For readability it also obtained the average weighted mean of 3.6 Acceptable in general and for usability it obtained the average weighted mean of 3.4 Acceptable in general and it also obtained the average weighted mean of 3.5 of all variables and was rated Acceptable in general. This implies that students have an optimistic response towards WSL. A modified syntax for better readability can also be introduced to enhance more the quality of the WSL.

KEYWORDS: *ISO 9126 , waray scripting language, mother-tongue based scripting.*

1. INTRODUCTION

Eastern Samar State University in Borongan City is an educational institution with its primary goal is to produce excellent graduates in all fields of study offered in this university. Students enrolled in computer related programs in this university are using scripting language especially the first year college students. It is a flexible language that provides fast program execution. This study would like to developed a scripting language in Waray. Waray is one of the major language spoken in the Philippines, mostly in the Eastern Visayas Region. In this study, WSL a mother-tongue based scripting laguage is used to serve as a tool for first year college students enrolled in computer related program in Learning Programming easily.

A scripting language is a programming language that employs a hig-level construct to intepret and execute one command at a time. In general, scripting languages are easier to learn and faster to code more than structured and compiled languages such as C and C++ Magaret Rouse, 2016[4]. There are lots of scripting languages that are available worldwide, this scripting languages varies according to purpose. Every scripting language has its own set of words (keywords) and syntaxes used that start a computer to perform certain tasks there are some scripting languages around the world that uses their own dialect as keywords for easy understanding of the locals.

Most of all, the popular scripting languages that are being used today are in English language. There are articles that attempted to explain instructional laguage issues. Fischer and Perez, 2008[1] said that students may struggle to learn the content in their second language if their academic knowledge is not sufficiently strong as their first language. That's why many people are having difficulties in learning scripting especially to those non-native speakers of English due to their lack of knowledge in English language. This situation leads to the possible problems of understanding and remembering the keywords because of the language used by the scripting languages used in teaching programming on other countries.

Programmers in the four corners of the world developed programming languages that could help learn programming easier. They wrote programming languages using their own national languages or dialects. Leon Lukaszewics, 1961[3] developed a Polish programming language called SAKO. According to Jim Cummins, 1991[2]. To accomplish successful teaching and learning, teachers is require to use students' native language for instruction. The purpose of this study was to develop a Waray Scripting Language (WSL) a mother-tongue based scripting language that will help and enhance the programming skills of the students in the university.

2. READABILITY VS. WRITABILITY

2.1 Readability

Readability simply means that the programming language is easy to read and understand by the programmers because the language construct is nearly same as natural human language. This criterion was used as a gauge to evaluate the quality of WSL.

2.2 Writability

Writability means that it is easy and fast to create programs in that language because the language construct has minimal symbols which does not require many statements and focuses on simplification of code (concise).

Readability and Writability contradicts each other, a readable programming language does not always mean it is writable (vice versa). Being a readable language means that you need to write extra or sometimes unnecessary code to make it more understandable, but on the other hand if the languages omit these codes (to make writable easy and fast) programs will be obscure and reader needs an additional documentation to understand the code. Readability and writability should not be biased on the level of skills of the person who reads and writes the code. Non-programmers and beginners might find it difficult to read and write on that particular programming language, but experienced programmers on that language will find it easy. Readability and writability should be based on syntax (language constructs) and semantics meaning and how it is understood.(https://williamarchibaldspooner.wordpress.com)

2.3 Code Readability

In the study of Buse & Weimer, 2010[5], Code Readability can be defined as a human judgment on how easy it is to understand a program source code. Further more, by Yahya Tashtous and Izzat Alsmadi, 2013[6] , although readable code is less erroneous, more reusable and code readability is not easy to measure by a deterministic function same as maintainability, reliability and reusability. However, a source code features that affect readers' understanding of an existing source code. Studying the impact of programming Features on Code Readability.

3. METHODOLOGY

This chapter covers the methods and steps used in developing this scripting language. This part also discusses the development model used in this study.

3.1 Software Development

In developing the scripting language, this study used racket programming language because racket was designed to create other programming languages. Racket helps programmers create and quickly deploy new languages.

In designing the syntax, this study consulted a linguist on what appropriate Waray words to be used as keywords in WSL (Waray Scripting Language).

3.2 Software Development Model

This study used prototyping model in the development of the Waray scripting language. The prototyping Model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed. This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and error process that takes place between the developers and the users.

The Prototype Model was used in developing this programming language to achieve he requirements and to meet the objectives of this study by trial and error process until a good prototype is achieved.

Prototype Model

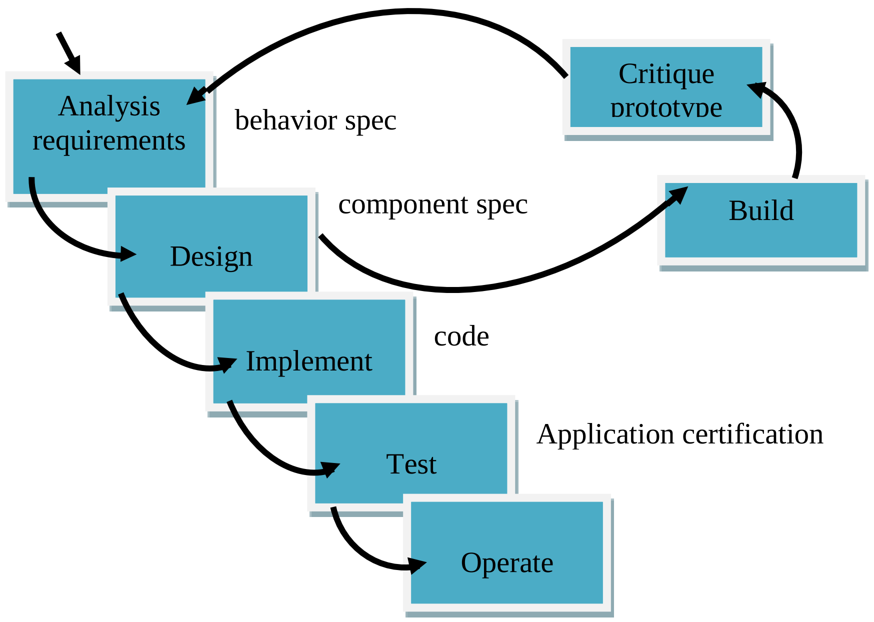


Figure 1. Prototyping Model

3.3 Software Employed

WSL was developed using Racket programming language. This language is free and open-source that were designed to develop other programming languages.

3.4 Hardware Employed

The following hardware specification is the minimum requirement to run and use WSL successfully:

- Processor Intel(R) Core(TM)2 Duo CPU 1.67GHz
- Ram 2.00 GB
- Hard Disk 500 GB
- Display 1280x800 (32bit) (60Hz)

3.5 WSL KEYWORDS AND SYNTAXES

Table 1 shows the list of keywords of WSL including the syntaxes employed per keyword. These syntaxes served as guide for programmers in using WSL and running an error-free program.

Table 1. Keywords and Syntaxes used in WSL

Keywords	Description	Syntax
Isurat-an	Displays strings or integers	(Isurat-an *stx*)
Ideklara-an	Declares a variable	(Ideklara-an <id> <expr>) (Ideklara-an (<id> <id>*) <expr>+)
Dugangi-an	Add numbers	(dugangi-an <num> hin <num>*)
Ibani-an	Subtract numbers	(ibani-an <num> hin <num>*)
Dobleha-an	Multiply numbers	(dobleha-an <num> hin <num>*)
Tungaa-an	Divide numbers	(Tungaa-an <num> hin <num>)
Mamadako-an	Greater than	(Mamadako-an <num> hin <num>)
Mamaguti-an	Less than	(Mamaguti-an <num> hit <num>)
Maadako-o-parehas-an	Greater than or equal	(Mamadako-o-parehas-an <num> hit <num>)
Diri-parehas-an	Not Equal	(Diri-parehas-an <num> hit <num>)
Mamaguti-o-parehas-an	Less than or equal	(Mamaguti-o-parehas-an <num> hit <num>)
Kun…Kun-diri	If…else	(kun (<condition>) true-expr kun-diri false-expr)

Figure 2. shows the process of code flow for WSL during runtime including their explanation.

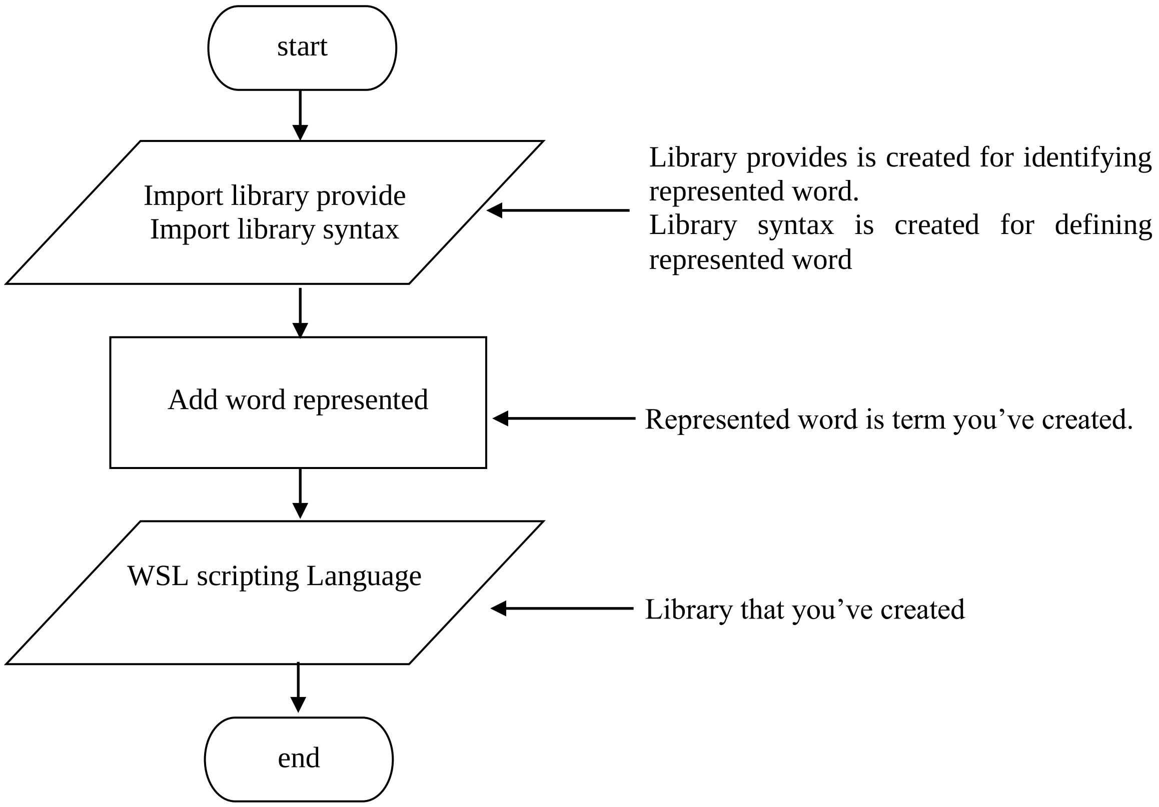

start
Import library provide
Import library syntax
Library provides is created for identifying represented word.
Library syntax is created for defining represented word
Add word represented
Represented word is term you've created.
WSL scripting Language
Library that you've created
end

Figure 2. The basic code flow for (WSL) during runtime.

Evaluation

Research Design

This study used a developmental-evaluative research design in which the focus was more on the impact of the WSL as a tool for the first year college students in Eastern Samar State University in their learning in programming subject using scripting language. WSL was subjected for quality evaluation in terms of (1) Usability, (2) Readability, and (3) Writability.

Data Gathering and Instrumentation

This study used questionnaire in collecting data from the respondents of the study. The questionnaire contained two levels of questions the first level is all about software development. The last level of question consists of three factors that respond to the performance of the study these are usability, readability, and writability. Each factor has four elements and the respondents were instructed on how to answer each level of questions. Each question was rated the following Likert Scale: (5) strongly Agree, (4) Agree, (3) Neither agree or disagree, (2) Disagree, and (1) Strongly disagree.

Data Analysis

This study used questionnaires to obtain data and information from the respondents concerning the performance of WSL in terms of usability, readability, and writability. Questionnaires responses were answered. Tabulated and analyzed using Frequency and Weighted Mean. Tabulated data were interpreted using table 3 below.

Table 3. Interpreted Scale

Scale	Interpretation
4.20 – 5.0	Strongly Acceptable
3.40 – 4.10	Acceptable
2.60 – 3.30	Neither Acceptable nor unacceptable
1.80 – 2.50	Unacceptable
1.00 – 1.70	Strongly unacceptable

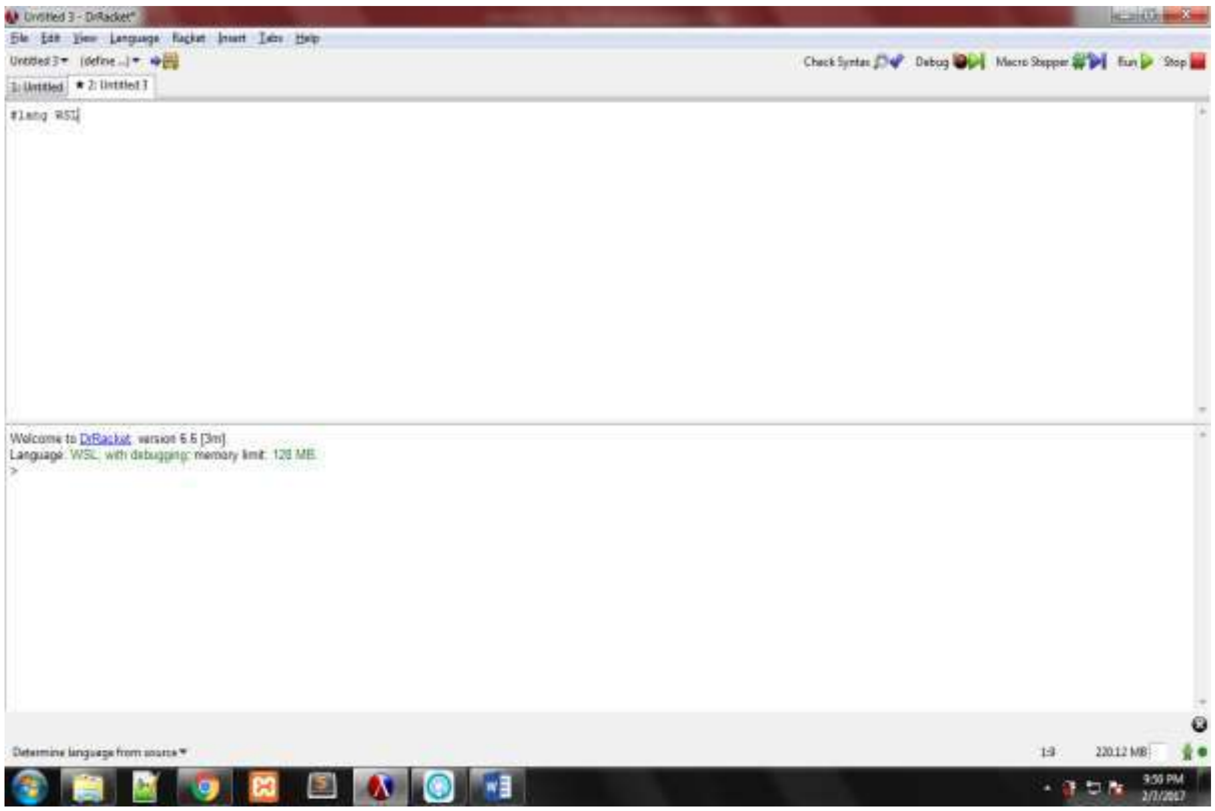
4.1 RESULTAND DICUSSION

This Chapter shows the result of the development process and the result of the evaluation on the quality and usability of WSL.

Software Development

Waray Scripting Language (WSL) User Interface

WSL was designed using Rackect, an open-source programming language that is intended to design new other languages. This study created a library under Racket and imported this to enable the execution of WSL codes in DrRacket interface. Figure 4 shows the DrRacket interface with library "#lang WSL". This "#lang WSL" should be the first line of WSL code to allow end-users to run the compiled scripting.

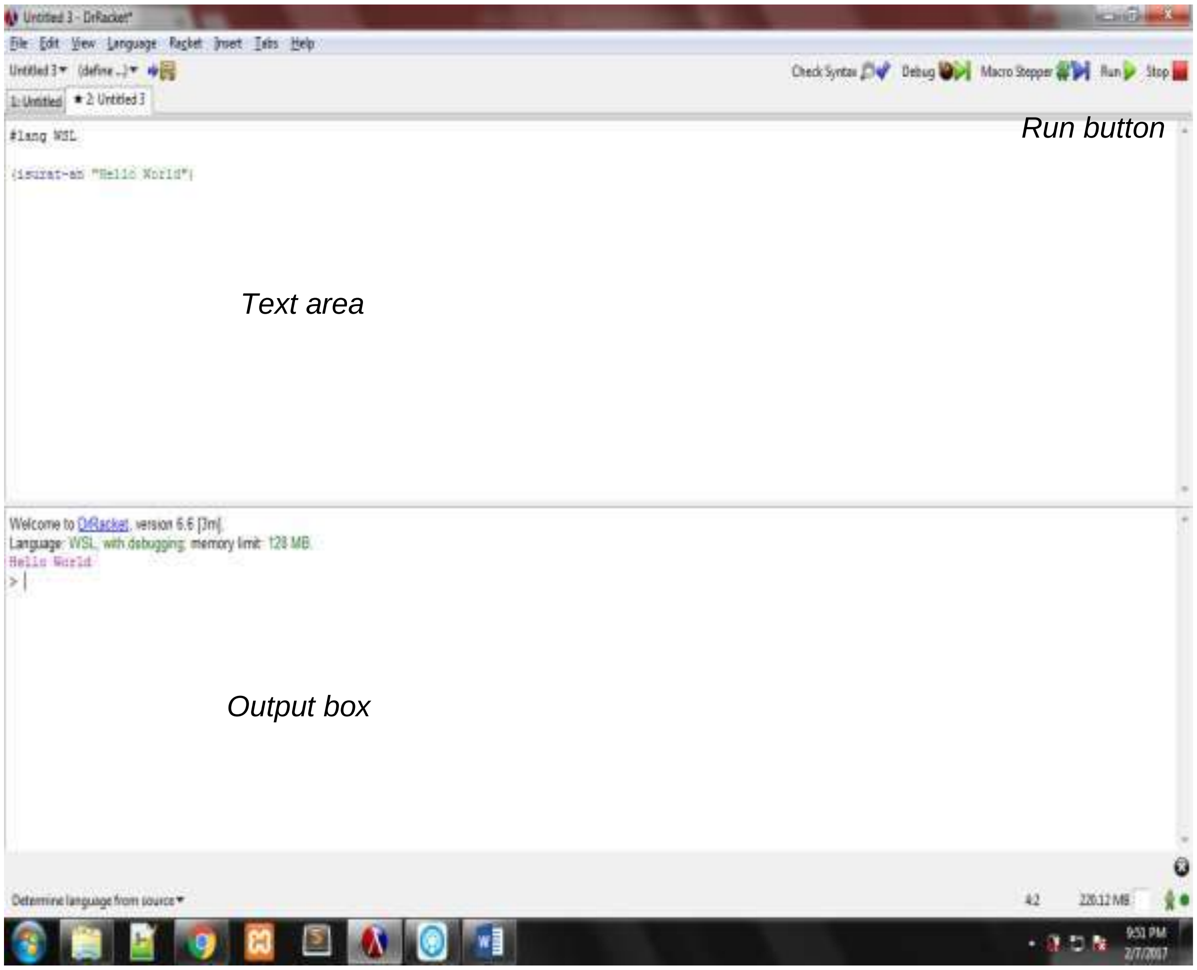

Untitled 3 - DrRacket*
File Edit View Language Racket Insert Tabs Help
Untitled 3 (define ...)
Check Syntax Debug Macro Stepper Run Stop
1: Untitled ★ 2: Untitled 3
Determine language from source
1:9 220.12 MB
9:50 PM 2/7/2017

```
#lang WSL
```

```
Welcome to DrRacket, version 6.6 [3m].
Language: WSL, with debugging; memory limit: 128 MB.
>
```

Figure 4. DrRacket – WSL User Interface

The user will type WSL codes in the text area and click run button on the upper-right corner to execute the program. The output of the program is seen in the output box below the screen. Figure 5 shows the WSL code with a running program.


Untitled 3 - DrRacket*
File Edit View Language Racket Insert Tabs Help
Untitled 3 (define ...)
1: Untitled ★ 2: Untitled 3
Check Syntax Debug Macro Stepper Run Stop
Run button
Text area
Output box
Determine language from source
4:2 220.12 MB

```
#lang WSL

(isurat-an "Hello World")
```

```
Welcome to DrRacket, version 6.6 [3m].
Language: WSL, with debugging; memory limit: 128 MB.
Hello World
> |
```

Figure 5. Code with Running Syntax

Result

The data from the evaluation were calculated and analyzed through the use of appropriate statistical techniques. 141 respondents were made to answer the questionnaire to verify the quality of WSL in terms of usability, readability and writability.

Table 4. Performance Level of the Proposed WSL: Waray Scripting Language in terms of Usability.

CRITERIA	Responses (f)					(N=141) Weighted mean	Interpretation
Usability	1	2	3	4	5		
a. Performance	0	0				3.5	Acceptable
b. Solution	0	0				3.5	Acceptable
c. Actual output	0	0			0	3.3	Neutral
Average Weighted Mean						3.4	Acceptable

Table 4 shows the result of the evaluation in terms of usability. The respondents gave 3.5 or acceptable rating under performance, 3.5 or acceptable for solution and 3.3 or neither acceptable nor unacceptable for the actual output. Having an average rate of 3.4 interpreted as acceptable.

Table 5. Performance Level of the Proposed WSL: Waray Scripting Language in terms of Readability.

CRITERIA	Responses (f)					(N=141) Weighted mean	Interpretation
Readability	1	2	3	4	5		
a. Performance	0	2				3.7	Acceptable
b. Solution	0	3				3.4	Acceptable
c. Actual output	0	2				3.7	Acceptable
Average Weighted Mean						3.6	Acceptable

Table 5 shows the performance of the proposed study under the criterion Readability. The respondents gave 3.7 **acceptable** under performance, 3.4 **acceptable** for solution and 3.7 **acceptable** for the actual output; with an average result of 3.6 interpreted as **acceptable** indicates that more than a half of respondents agreed that WSL conforms to its readable quality.

Table 6. Performance Level of the Proposed WSL: Waray Scripting Language in terms of Writability.

CRITERIA	Responses (f)					(N=141) Weighted mean	Interpretation
Writability	1	2	3	4	5		
a. Performance	0	2				3.4	Acceptable
b. Solution	0	3				3.6	Acceptable
c. Actual output	0	2				3.6	Acceptable
Average Weighted Mean						3.5	Acceptable

Table 6 shows the performance of the Proposed study under the criterion Writability. The respondents gave 3.4 **acceptable** under performance, 3.6 **acceptable** for solution and 3.6 **acceptable** for the actual output. WSL writability quality earned n average weighted mean of 3.5 interpreted as **acceptable**. The result indicates that WSL conforms to the writability standard of programming languages.

Table 7. Summary of Performance of the Proposed WSL: Waray Scripting Language

CRITERIA	Weighted Mean	Interpretation
Usability	3.4	Acceptable
Readability	3.6	Acceptable
Writability	3.5	Acceptable
Overall Weighted Mean	3.5	Acceptable

Table 7 shows the summary of WSL: Waray Scripting Language quality evaluation results. The Language earned an overall weighted mean of 3.5 interpreted as **acceptable.** This overall result indicates that WSL adheres to the quality of a scripting language.

This result indicates that WSL is of usable quality and can be used for learning programming and creating programming applications.

5. SUMMARY

Summary

This study was conducted to develop a scripting language that uses waray dialect as keywords and evaluate its performance. The purpose of developing WSL or Waray Scripting Language is to introduce programming to beginners here in Eastern Samar State University. WSL or Waray Scripting Language can only construct basic codes because it is design for a beginner which is first year college students in Eastern Samar. The syntax of WSL (Waray Scripting Language) is much more different than C programming language, the syntax of WSL is parenthesized and verbose unlike C and other programming language that uses symbols.

REFERENCES

1. *Fischer and Perez, 2008. Understanding English Through Mathematics*
2. *Jim Cummins, 1991. "A Conceptual framework of Bilingual Special Education Teacher Program".*
3. *Leon Łukaszewicz, 1961. "System automatycznego kodowania SAKO"*
4. *Magaret Rouse, 2001. What scripting languages should every DevOp know?*
5. *Raymond P.L. Buse & Westley Weimer, 2010. Learning a Metric for Code Readability.*
6. *Yahya Tashtous and Izzat Alsmadi, 2013. Impact of Programming Features on Code Readability*

APPENDIX A
(WSL CODE)

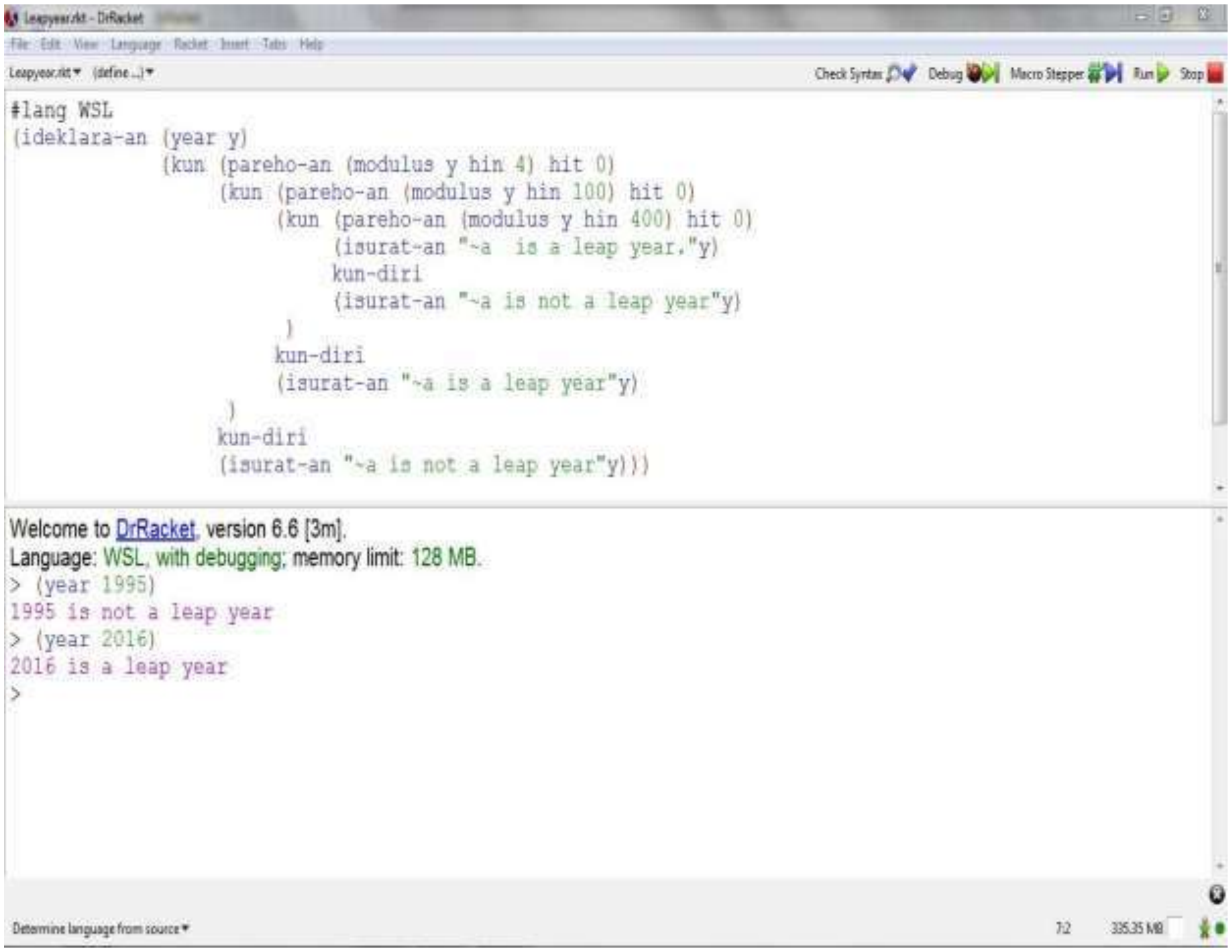
```
#lang racket
 ( provide#%top-interaction
   #%app
   #%datum
   #%top
   #%module-begin
   )
  ( provide const)
 ( struct const(v)#:transparent)

#|Conditions
 Kun diri condition |#
 ( provide kun)
 ( define-syntax (kun stx)
   ( syntax-case stx (kun-diri)
     [(_condition true-expr kun-diri false-expr)
       #'( cond [condition true-expr]
              [ else false-expr])]
      [(_conditon true-expr)#'( cond [condition true-expr])]))

 #| Print: Displays strings or integers
  Isurat-an |#
   ( provide isurat-an)
   ( define-syntax isurat-an
   ( syntax-rule ()
      [( isurat stx) ( printf stx)]
    [( isurat (void))( printf (void))]
    [( isurat stx1 stx2…)(printf stx1 stx2…)]))

#| Print: Displays strings or integers next line
 Isurat-in |#
 ( provide isurat-in)
 ( define-syntax isurat-in
    ( syntax-rules()
    [( isurat stx)( displayin (stx)]
    [( isurat (void))(displayin (void))]
```

```
     [( isurat stx1 stx2…)( displayin stx1 stx2…)]))

#| Declares Variable/Identifiers
 Ideklara-an |#
 ( provide ideklara-an)
 ( define-syntax ideklara-an
   ( syntax-rules ()
     [( ideklara-an  id expr)(define id expr)]
     [( ideklara-an (id1 id2…) expr…)( define (id1 id2…) expr…)]))


#| Operators:
 Add = dugangi-an
 Subtract = ibani-an
 Multiply = dobleha-an
 Divide = tungaa-an

|#
 ( provide dugangi-an)
 ( provide dobleha-an)
 ( provide ibani-an)
   ( provide tungaa-an)

#| Addition |#
 ( define-syntax dugangi-an
 ( syntax-rules (hin)
    ([ dugangi-an n1 hin n2…][ + n1 n2…])))

#| Subtraction |#
 ( define-syntax ibani-an
 ( syntax-rules (hin)
    ([ ibani-an n1 hin n2…][ - n1 n2…])))

#| Modulus |#
 ( define-syntax modulus
 ( syntax-rules (hin)
    ([ modulus n1 hin n2…][ modulo  n1 n2…])))

#| Multiplication |#
 ( define-syntax dobleha-an
 ( syntax-rules (hin)
    ([ dobleha-an n1 hin n2…][ * n1 n2…])))

#| Division |#
 ( define-syntax tungaa-an
 ( syntax-rules (hin)
    ([ tungaa-an n1 hin n2…][ / n1 n2…])))

   ( provide quot)
   ( define-syntax quot
   ( syntax-rules (hin)
    ([ quot  n1 hin n2…][ quotient n1 n2…])))

;(struct dugangi-an (n1 n2…) #: transparent)
;(struct dobleha-an (e1 e2…) #: transparent)
```

```
;(struct ibani-an (e1 e2…) #: transparent)
;(struct tungaa-an (e1 e2…) #: transparent)
;( define (wpl-eval e)
; ( match e
  ; [( dugangi-an n1 n2…) (+ n1 n2…)]
  ; [( ibani-an e1 e2…) (- e1 e2…)]
  ; [( dobleha-an e1 e2…) (* e1 e2…)]
  ; [( tungaa-an e1 e2…) (/ n1 n2…)]

#| Conditional Operators
 >= mamadako-an
 <= mamaguti-an
 '=' = pareho-an
   '!=' = diri-parehas-an
 <= = mamaguti-o-parehas-an
 >= = mamadako-o-parehas-an

|#
   ( provide mamadako-an)
   ( define-syntax mamadako-an
    ( syntax-rules (hit)
      ([ quot n1 hit n2][ > n1 n2 ])))

    ( provide mamaguti-an)
    ( define-syntax mamaguti-an
    ( syntax-rules (hit)
      ([ quot n1 hit n2][ < n1 n2 ])))

  ( provide pareho?)
    ( define-synta-rule (pareho? stx1 stx2) ( equal? stx1stx2 ))
  ( provide pareho-an)
  ( define-syntax pareho-an
    ( syntax-rules (hit)
      [( pareho-an n1 hit n2 )( = n1 n2 )]
      [( pareho-an string?1 hit string?2)( = string?1 string?2 )]))

( provide mamadako-oparehas-an)
  ( define-syntax mamadako-oparehas-an
    ( syntax-rules (hit)
      [( quot n1 hit n2][ >= n1 n2 ])))


( provide mamaguti-oparehas-an)
  ( define-syntax mamaguti-oparehas-an
    ( syntax-rules (hit)
      [( quot n1 hit n2][ <= n1 n2 ])))

( provide sunod-nga-linya)
( define-syntax-rule ( sunod-nga-linya) (newline))

( provide o)
( define-syntax ( o x y)
( let ([z x])
( if z z y )))
( provide nagan )
( define-syntax ngan
   ( syntax-rules ()
    ((_) #t)
    ((_ e) e)
    ((_ e1 e2…)
    ( if e1
  ( my-and e2…)        #f))))
;Looping statements
( provide let)
( provide for)

( provide while)
( define-syntax while
   ( syntax-rules ()
    ((_ pred b1…)
    ( let loop () (when pred b1…( loop))))))

( define-syntax for
( syntax-rules ()
   ((_ ( I from to) b1…)
     ( let loop(( I from ))
      ( when ( < I to )
              b1…
              ( loop( + i 1 )))))))

;( define-syntax isurat-an
  ;( display stx ))

;( provide ibani-an )
;( define-syntax-rule ( ibani-an n1 hin n2…)( - n1 n2…))

;( provide dobleha-an )
;( define-syntax-rule ( dobleha-an n1 hin n2…)( * n1 n2))

;( provide tungaa-an )
;( define-syntax-rule ( tungaa-an n1 hin n2…)( / n1 n2))
```

APPENDIX B
WSL SAMPLE PROGRAM CODES

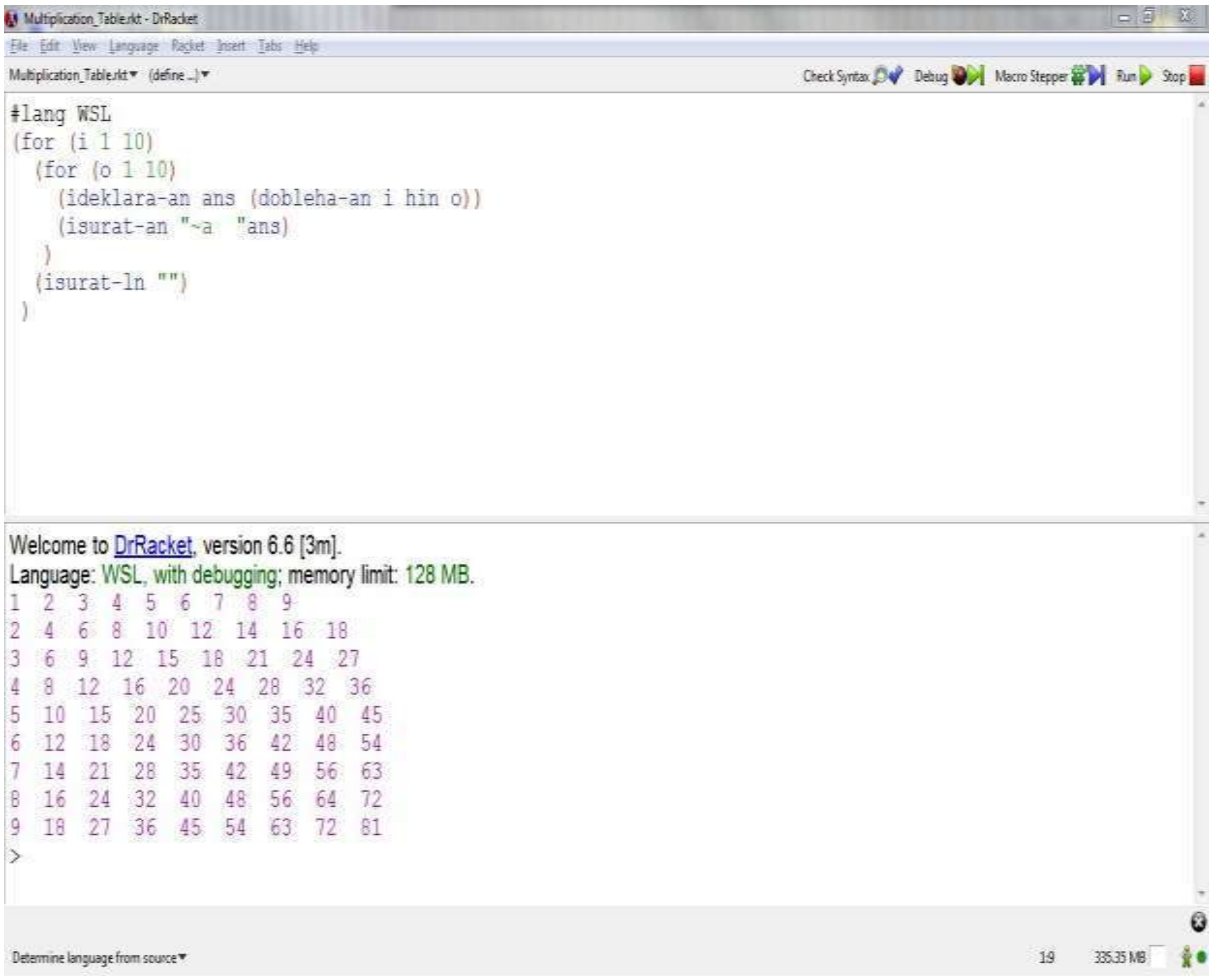
Leap Year



```
#lang WSL
(ideklara-an (year y)
             (kun (pareho-an (modulus y hin 4) hit 0)
                  (kun (pareho-an (modulus y hin 100) hit 0)
                       (kun (pareho-an (modulus y hin 400) hit 0)
                            (isurat-an "~a  is a leap year."y)
                            kun-diri
                            (isurat-an "~a is not a leap year"y)
                        )
                       kun-diri
                       (isurat-an "~a is a leap year"y)
                   )
                  kun-diri
                  (isurat-an "~a is not a leap year"y)))
```

```
Welcome to DrRacket, version 6.6 [3m].
Language: WSL, with debugging; memory limit: 128 MB.
> (year 1995)
1995 is not a leap year
> (year 2016)
2016 is a leap year
>
```

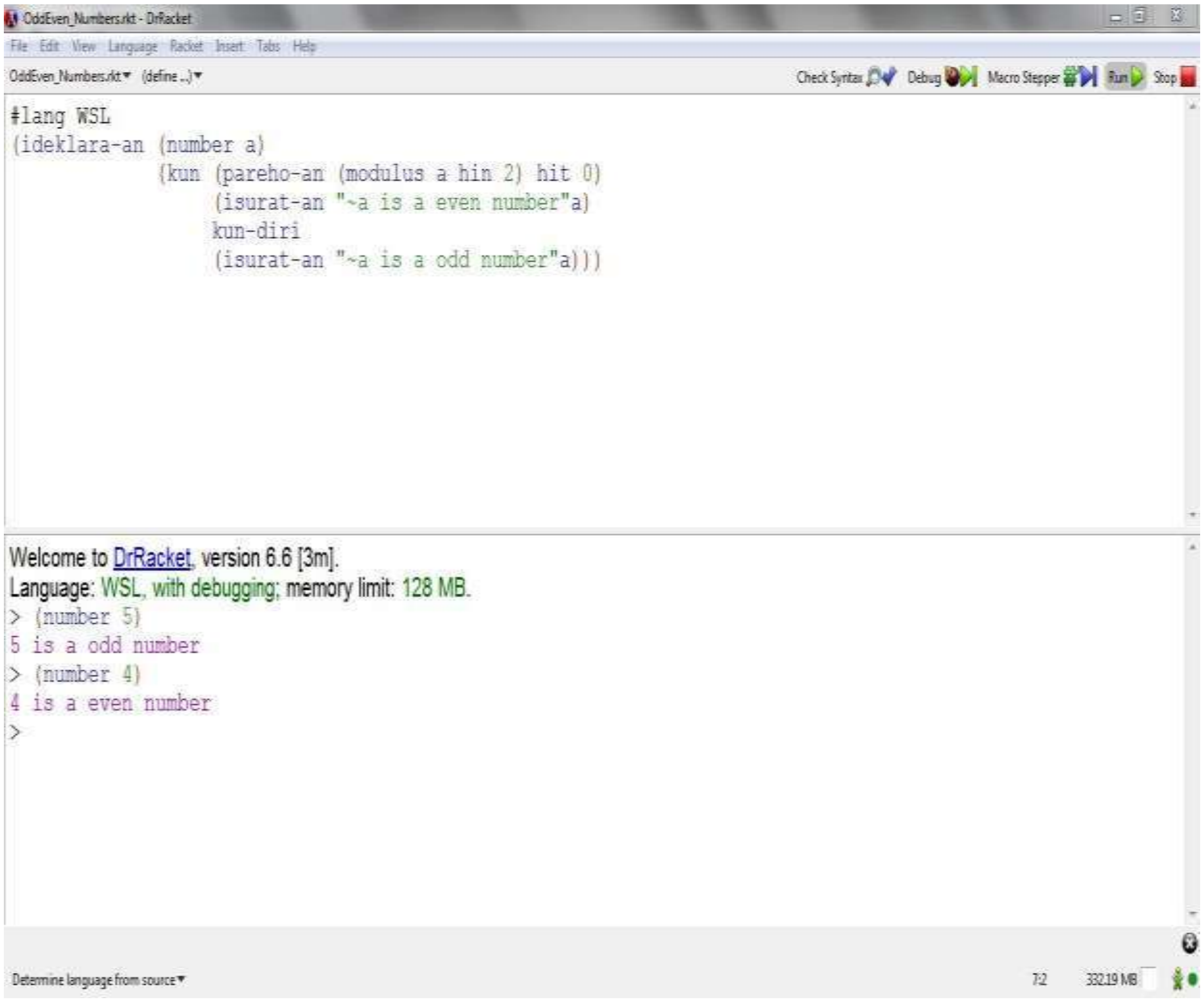
Multiplication Table


Multiplication_Table.rkt - DrRacket
File Edit View Language Racket Insert Tabs Help
Multiplication_Table.rkt (define ...)
Check Syntax Debug Macro Stepper Run Stop
Determine language from source
19 335.35 MB

```
#lang WSL
(for (i 1 10)
  (for (o 1 10)
    (ideklara-an ans (dobleha-an i hin o))
    (isurat-an "~a  "ans)
   )
  (isurat-ln "")
 )
```

```
Welcome to DrRacket, version 6.6 [3m].
Language: WSL, with debugging; memory limit: 128 MB.
1  2  3  4  5  6  7  8  9
2  4  6  8  10  12  14  16  18
3  6  9  12  15  18  21  24  27
4  8  12  16  20  24  28  32  36
5  10  15  20  25  30  35  40  45
6  12  18  24  30  36  42  48  54
7  14  21  28  35  42  49  56  63
8  16  24  32  40  48  56  64  72
9  18  27  36  45  54  63  72  81
>
```

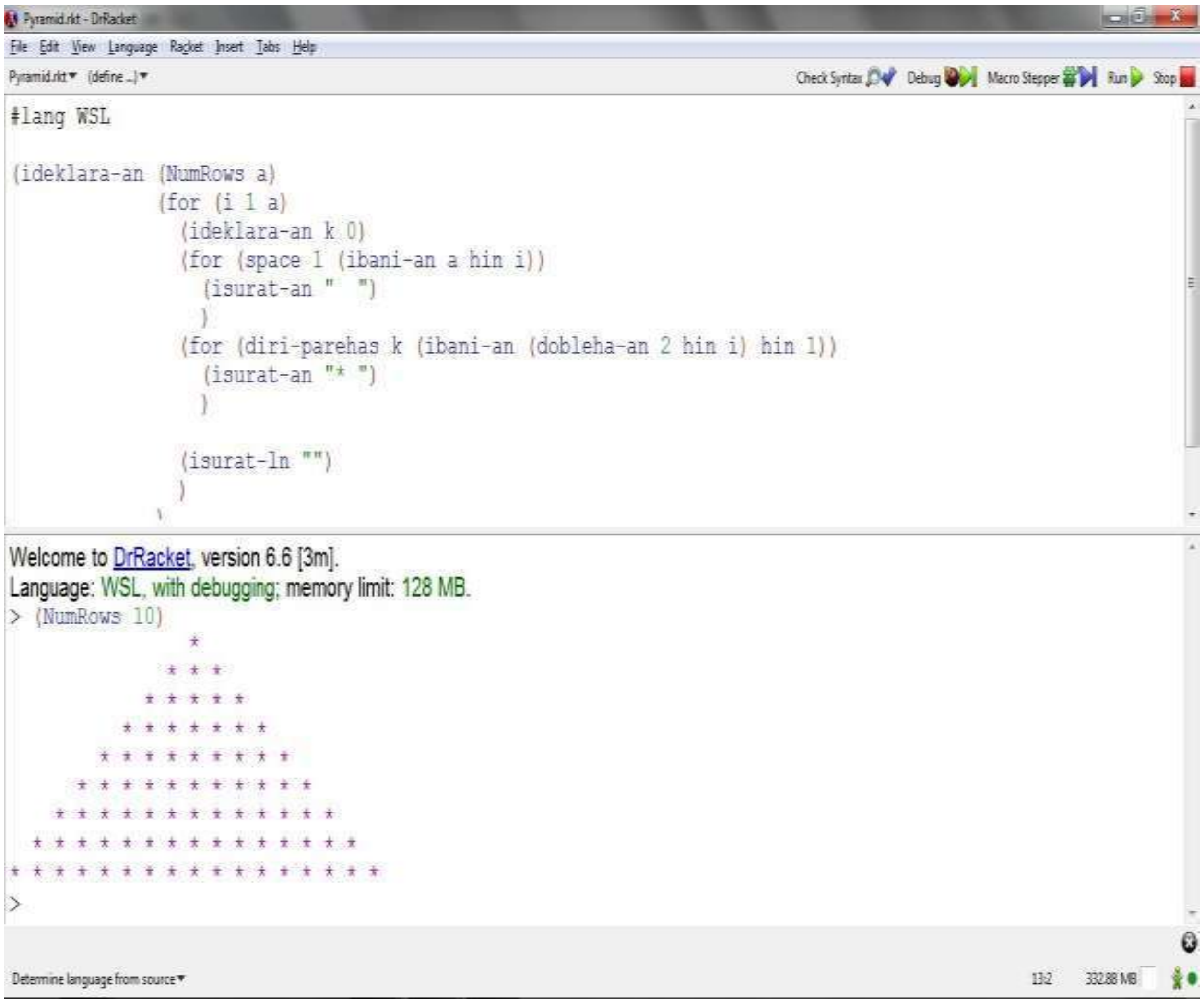
Odd Even Number


OddEven_Numbers.rkt - DrRacket
File Edit View Language Racket Insert Tabs Help
OddEven_Numbers.rkt (define ...)
Check Syntax Debug Macro Stepper Run Stop
Determine language from source
7:2
332.19 MB

```
#lang WSL
(ideklara-an (number a)
            (kun (pareho-an (modulus a hin 2) hit 0)
                 (isurat-an "~a is a even number"a)
                 kun-diri
                 (isurat-an "~a is a odd number"a)))
```

```
Welcome to DrRacket, version 6.6 [3m].
Language: WSL, with debugging; memory limit: 128 MB.
> (number 5)
5 is a odd number
> (number 4)
4 is a even number
>
```

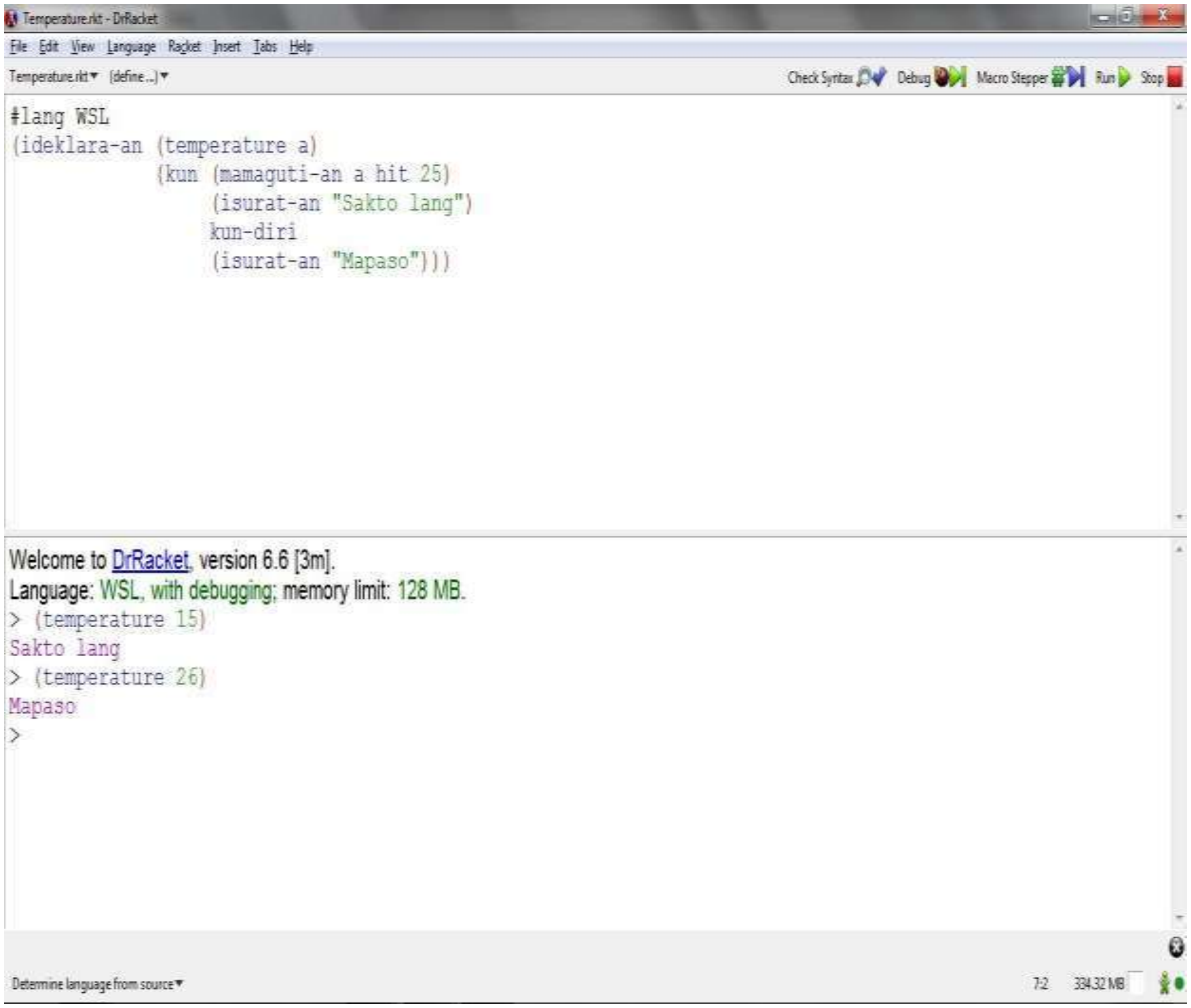
Pyramid



```
#lang WSL

(ideklara-an (NumRows a)
             (for (i 1 a)
               (ideklara-an k 0)
               (for (space 1 (ibani-an a hin i))
                 (isurat-an "  ")
                 )
               (for (diri-parehas k (ibani-an (dobleha-an 2 hin i) hin 1))
                 (isurat-an "* ")
                 )

               (isurat-ln "")
               )
             )
```

```
Welcome to DrRacket, version 6.6 [3m].
Language: WSL, with debugging; memory limit: 128 MB.
> (NumRows 10)
                  *
                * * *
              * * * * *
            * * * * * * *
          * * * * * * * * *
        * * * * * * * * * * *
      * * * * * * * * * * * * *
    * * * * * * * * * * * * * * *
  * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * *
>
```

Temperature


Temperature.rkt - DrRacket
File Edit View Language Racket Insert Tabs Help
Temperature.rkt (define ...)
Check Syntax Debug Macro Stepper Run Stop
Determine language from source
7:2 334.32 MB

```
#lang WSL
(ideklara-an (temperature a)
             (kun (mamaguti-an a hit 25)
                  (isurat-an "Sakto lang")
                  kun-diri
                  (isurat-an "Mapaso")))
```

```
Welcome to DrRacket, version 6.6 [3m].
Language: WSL, with debugging; memory limit: 128 MB.
> (temperature 15)
Sakto lang
> (temperature 26)
Mapaso
>
```