Tcl Reference Guide
For Beginners

Table of Contents:

1. Hello World
2. Tcl Comments
3. Constants, Variable, Basic Operation
4. Mathematical Expression
5. Calculation
6. Operators: Arithematic
7. Control Statements : Conditional Statements
8. If, Else if, if then , switch statments
9. Looping Structure: For Loop, While Loop
10. Procedures
11. Array and Strings
12. Package in Tcl
13. List in Tcl
14. File Handling in Tcl
15. Appending Lists and List Functions
16. File Handling functions

No. 1 – Hello World

Example

```
puts "Hello world"
```

Output

```
Hello world
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 2 – Tcl Comments

Example

```
puts "Hello world"
```

#hello world program

Output

```
Hello world
```

Remarks

Compiler don't compile Comments, which are started with #

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 3 – Constant

Example

abc is same as "abc"
"5" is same as 5

Output

<Output of the above example goes here>

Remarks

Constant are stored by the variable on the memory space

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 4 – Variable

Example

```
set a 5
puts $a    #it will print out 5
```

Output

```
5
```

Remarks

Variable a stores the vale 5 and $a gives the address of the variable a.

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 5 – Basic operation

Example

```
set income 5000
puts "income is $income"    #$ get the value of the variable
```

Output

```
income is 5000
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 6 – Mathematical Expression (expr)

Example

```
set a 5.0              #set  value of a as 5.0
expr $a +5             # value of a is add with 5
expr int($a/3)              #a is divided with 3 and save as integer
```

Output

```
10
1
```

Remarks

`expr` is used for mathematical operation or arithmetic operation

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 7 – Mathematical Expression (expr) II

Example

```
set variableA "10"
set result [expr $variableA / 9];
puts $result
set result [expr $variableA / 9.0];
puts $result
set variableA "10.0"
set result [expr $variableA / 9];
puts $result
```

Output

```
1
1.1111111111111112
1.1111111111111112
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 8 – Calculation

Example

```
set a [expr 5+6]                #result will set to a
set b [SomeFunction 6]          #will set the "return value" of
puts $a                         #"SomeFunction to b
#puts $b
```

Output

```
11
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 9– Operators: Arithmetic I

Example

```
set a [expr 5+6]
set b [expr 6/3]
set c [a+b]
puts c
```

Output

```
13
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 10 – Operators: Arithmetic II

Example

```
set a [expr 5+6]                #result will set to a
set b [SomeFunction 6]          #will set the "return value" of
puts $a                         #"SomeFunction to b
#puts $b
```

Output

```
11
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 11 – Control Statement: A Simple Conditional

Example

```
set vbl 1
if {$vbl == 1} { puts "vbl is one" }
```

Output

```
vbl is one
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 12 – Conditional Statement: else clause

Example

```
if {$vbl == 1} {
puts "vbl is one"
} else {
puts "vbl is not one"
}
```

Output

```
11
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 13 – Conditional Statement: elseif clause

Example

```
if {$vbl == 1} {
        puts "vbl is one"
     } elseif {$vbl == 2} {
        puts "vbl is two"
     } else {
        puts "vbl is not one or two"
     }
```

Output

```
11
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 14 – Conditional Statement: if then clause

Example

```
if {
        $vbl == 1 || $vbl == 2 || $vbl == 3
    } then {
        puts "vbl is one, two or three"
    }
```

Output

```
11
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 15 – Conditional Statement : Switch

Example

```
switch xyz {
        a  -
        b {
           # Correct Comment Placement
           expr 1
        }
        c {
           expr 2
        }
        default {
           expr 3
        }
    }
```

Output

```
11
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 16– Looping - For

Example

```
for {set a 0} {$a <= 10} {incr a 1} {
   puts "a is $a"
   }
```

Output

```
a is 1
a is 2
a is 3
a is 4
a is 5
a is 6
a is 7
a is 8
a is 9
a is 10
```

Remarks

Unlike the 'C' language, "for" is a command not a statement. Therefore, be careful to put the curly braces exactly as shown. The "for" command takes three inputs contained in the curly braces. Also, note the spaces between the braces.

Reference

- http://www.fundza.com/tcl/quickref_1/#for

No. 17 – Looping-While

Example

```
set lineCount 0
      while {[gets $chan line] >= 0} {
          puts "[incr lineCount]: $line"
      }
```

Output

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 18 – Procedures

Example

```
proc helloWorld {} {
   puts "Hello, World!"
}
helloWorld
```

Output

```
Hello,World!
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 19 – String I

Example

```
set myVariable hello
puts $myVariable
```

Output

```
hello
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 20 – string II

Example

```
set s1 “Hello World”
puts “uppercase string of s1”
puts [string toupper $s1]
puts “lowercase string of s1”
puts [string tolower $s1]
```

Output

```
uppercase string of s1
HELLO WORLD
lowercase string of s1
hello world
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 21 –String III (Append Command)

Example

```
set s1 "Hello"
append s1 "World"
puts $s1
```

Output

```
Hello World
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 22 – packages

Example

```
puts "Hello world"
```

Output

```
Hello world
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 23 – Arrays I

Example

```
array set colorcount {
    red   1
    green 5
    blue  4
    white 9
}

foreach {color count} [array get colorcount] {
   puts "Color: $color Count: $count"
}
```

Output

```
    Color: blue Count: 4
    Color: white Count: 9
    Color: green Count: 5
    Color: red Count: 1
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 24 – Arrays II

Example

```
array set colorcount {
    red   1
    green 5
    blue  4
    white 9
}

foreach color [array names colorcount] {
   puts "Color: $color Count: $colorcount($color)"
}
```

Output

```
    Color: blue Count: 4
    Color: white Count: 9
    Color: green Count: 5
    Color: red Count: 1
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 25 – List I

Example

```
set colorList1 {red green blue}
set colorList2 [list red green blue]
set colorList3 [split "red_green_blue" _]
puts $colorList1
puts $colorList2
puts $colorList3
```

Output

```
red green blue
red green blue
red green blue
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 26 – List II (Lsearch)

Example

```
lsearch {a b c d e} c
lsearch -inline {a20 b35 c47} b*
lsearch -start 3 {a b c a b c} c
```

Output

```
2
b35
 5
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 27 – List III (Split)

Example

```
split "Hello world" {}
```

Output

```
H e l l o { } w o r l d
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 28– List IV (Lrange)

Example

```
% lrange {a b c d e} 0 1
% lrange {a b c d e} end-2 end
% lrange {a b c d e} 1 end-1
```

Output

```
a b
c d e
b c d
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 29 – File Handling

Example

```
set fp [open "input.txt" w+]
puts $fp "test"
close $fp
set fp [open "input.txt" r]
set file_data [read $fp]
puts $file_data
close $fp
```

Output

```
test
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 30– File Handling

Example

```
set fp [open "input.txt" w+]
puts $fp "test"
close $fp
set fp [open "input.txt" r]
set file_data [read $fp]
puts $file_data
close $fp
```

Output

```
test
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 31 – Reading user input

Example

```
puts "Please tell me your name."
gets stdin Name
puts "Hello, $Name!"
```

Output

```
puts "Please tell me your name."
gets stdin Name
puts "Hello, $Name!"
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

References

http://wiki.tcl.tk/11833

http://www.tutorialspoint.com/execute_tcl_online.php

No. 32 – Conditionals

Example

```
puts "Hey dude, how old might you be?"
gets stdin Age
if {$Age < 18} {
    puts "You are a child or a teen-ager"
} else {
    puts "You are an adult now"
}
```

Output

```
Hey dude, how old might you be?
23
You are an adult now
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

References

http://wiki.tcl.tk/11833

No. 33 – Adding Complexity

Example

```
puts "Hey dude, how old might you be?"
gets stdin Age
if {$Age < 12} {
    puts "You are a child"
} elseif {$Age < 19} {
    puts "You are a teen"
} else {
    puts "You are an adult now"
}
```

Output

```
test
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

References

<List all reference links/books here>

No. 34 – Continuing a procedure

Example

```
for {set x 1} {$x < 11} {incr x} {
if {$x == 5} {
puts " "
continue
}
puts "x = $x"
}
```

Output

```
x = 1
x = 2
x = 3
x = 4

x = 6
x = 7
x = 8
x = 9
x = 10
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

References

Tcl/Tk 8.5 Programming Cook Book Bert Wheeler

No. 35 – Breaking out of a procedure

Example

```
for {set x 1} {$x > 0} {incr x} {
if {$x == 5} {
puts "Upper limit reached"
break
}
puts "x = $x"
}
```

Output

```
x = 1
x = 2
x = 3
x = 4
Upper limit reached
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

References

Tcl/Tk 8.5 Programming Cook Book Bert Wheeler

No. 36 – Nested looping

Example : <saved as nest.tcl in bin>

```
if {$argc == 2} {
set x [lindex $argv 0]
set y [lindex $argv 1]
puts "Beginning the while loop"
for {set i $x} {$i <= $y} {incr i} {puts $i}
} else {
puts "Invalid number of arguments"
}
```

Output

```
% tclsh nest.tcl 1
Invalid number of arguments
```

And

```
% tclsh nest.tcl 5 10
5
6
7
8
9
10
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 37 – Joining two lists

Example

```
% concat {a b c} {1 2 3}
```

Output

```
a b c 1 2 3
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 38 – Joining list elements

Example

```
set input {{John Mary Bill} {Tom Fred Sally}}
{John Mary Bill} {Tom Fred Sally}
join $input
```

Output

```
John Mary Bill Tom Fred Sally
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 39 – Appending list elements

Example

```
set input {John Mary Bill}
#John Mary Bill
lappend input Tom
```

Output

```
John Mary Bill Tom
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 40 – Assigning list elements to variables

Example

```
lassign {John Mary Bill Tom Fred} 1 2 3
puts "$1 $2 $3"
```

Output

```
John Mary Bill
```

Remarks

Save this project in project_name.tcl in the local directory and run in tcl compiler from that directory.

Reference

- <Link to further details on this syntax (if any)>

- <Link to further examples using this syntax (if any)>

No. 41 – Retrieving an element from a list

Example

```
set input {John Mary Bill}
lindex $input 1
```

Output

```
John Mary Bill
Mary
```

Remarks

Run the First command first and then run the second command which gives the output of `Mary`

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 42 – Inserting elements into a list

Example

```
set input {John Mary Bill}
#John Mary Bill
set newinput [linsert $input 1 Tom]
#John Tom Mary Bill
puts $input
#John Mary Bill
puts $newinput
#John Tom Mary Bill
```

Output

```
John Mary Bill
John Tom Mary Bill
```

Remarks

Create project_name.tcl and inset the above example tcl program which on running returns the above output.

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 43 – Determining the number of elements

Example

```
llength {John Mary { Bill Tom }}
```

Output

```
3
```

Remarks

Run the above example example code on tcl compiler which gives the number of element in command.

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 44 – Getting a list element

Example

```
lrange {John Mary Bill Fred Tom Sally} 0 1
```

Output

```
John Mary
```

Remarks

Lrange gives the list element specified on the command

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 45 – Repeating elements

Example

```
lrepeat 3 a
```

Output

```
a a a
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 46 – Replacing elements

Example

```
lreplace {a b c d e} 1 1 X
```

Output

```
a X c d e
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>

- <Link to further examples using this syntax (if any)>

No. 47 – Reversing elements

Example

```
lreverse {a b c d e}
```

Output

```
e d c b a
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 48 – Searching a list

Example

```
lsearch -all {John Mary Bill John Mary Bill} Bill
```

Output

```
2 5
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 49 – Editing a list

Example

```
set input {John Mary Fred}
#John Mary Fred
lset input 1 Tom

#John Tom Fred
```

Output

```
John Mary Fred
John Tom Fred
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 50 – Sorting a list

Example

```
lsort -decreasing {a b c d e}
```

Output

```
e d c b a
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 51 – Splitting a string into a list

Example

```
split {John,Mary,Tom,Fred,Sally}
```

Output

```
John,Mary,Fred,Tom,Sally
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 52 – Opening a File

Example

```
set fp [open text.txt a+]
```

Output

```
file3174b0
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 53 – Configuring a file

Example

```
set fp [open text.txt r]
```

Output

```
file2f8d20
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 54 – Opening a command pipeline

Example

```
set fp [open "|cmd.exe /c dir text.txt" r]
```

Output

```
file2369eb0
```

Remarks

If the `open` command or one of the commands provided as arguments should return an error, a Tcl error will be generated when the close command is invoked on the channel unless the pipeline has been configured for non-blocking. If the channel is configured for non-blocking, no exit status will be returned.

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 55 – Writing a file

Example

```
set fp [open text.txt a]
```

Output

```
file2f81a0
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 56 – Reading a file

Example

```
set fp [open text.txt r]
```

Output

```
file31d780
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 57 – Reading a file-II

Example

```
read $fp
```

Output

```
Hello Again
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 58 – Closing a file

Example

```
% set fp [open text.txt a]
file5
% puts $fp "Hello Again"
% close $fp
```

Output

<Output of the above example goes here>

Remarks

The close command flushes the open channel of any pending data resulting in a write to disk and closes the channel. As you can see the close command has closed the file successfully as there were no errors returned.

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 59 – File handling

Background

Using the text editor of your choice create a text file containing the following text:
1,3,5,7,8,2,4,6,9
Save the file in your working directory as input.txt.

Example

```
# Check that a filename was provided
```

```
if { $argc>0 & $argv>0} {
# Assign the filename to a variable
set fname [lindex $argv 0]
# Open the file for read-only access
set fp [open $fname r]
# Read the contents of the file into a variable
set data [read $fp]
#Close the input file
close $fp
# Split the data and create a Tcl list
set input [split $data ","]
# Sort the list and load it into another list
set output [lsort -increasing $input]
# Open a file to write the data to
set fp [open output.txt w]
# Read through the list and write the data
foreach item $output {
puts $fp $item
}
#Close the file
close $fp
} else {
puts "No filename provided... Exiting Script"
exit
}
```

Command

```
tclsh filehandler.tcl input.txt
```

Output

```
1
2
3
4
5
6
7
8
9
```

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

No. 60 – <Syntax Name>

Example

<Illustrative short example using the syntax goes here> <make code font – courier new>

Output

<Output of the above example goes here>

Remarks

<Add comments on the implementation of the above syntax if any>

Reference

- <Link to further details on this syntax (if any)>
- <Link to further examples using this syntax (if any)>

References

<List all reference links/books here>