МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Компьютерные технологии в научных исследованиях

Лабораторная работа №1

Язык Tcl. Основные Команды. Ввод/вывод данных. Работа со списками и строками.

Запуск скрипта

Создание и запуск скрипта:

Вариант 2

```
touch script.tcl
subl script.tcl &
```

```
#!/usr/bin/tclsh

puts "Hello from Tcl!"
```

```
chmod +x ./script.tcl

tclsh ./script.tcl
```

Вывод данных на экран

Вывод строки на экран (в общем виде) :

```
puts ?-nonewline? ?channelid? string
```

Вывод строк на экран (с переводом строк):

```
puts "This is a first string"
puts "This is a second string"
```

Вывод строк на экран (без перевода строк):

```
puts –nonewline "This is a first string"
puts "This is a second string"
```

Вывод с явным указанием канала:

```
puts stdout "This is a string"
```

Работа с переменными: объявление переменных

Запись в общем виде:

```
set varName ?value?
```

Объявление переменных:

```
set var1 7
set var2 "This is a string"
```

Использование переменных:

```
set var3 $var2
# var3 = "This is a string"
```

Операторы группировки { } и " " (1)

Bash code:

```
var1=5
echo $var1

var2="This is a string"
echo $var1 $var2
```

Tcl code:

```
set var1 5
puts $var

set var2 "This is a string"
puts $var1 $var2 – ошибка!

puts "$var1 $var2" – нет ошибки
```

Операторы группировки { } и " " (2)

```
set var "This is a string"  ⟶  set var {This is a string}

puts $var
```

```
puts "$var"
```

```
puts {$var}
```

Чтение данных с клавиатуры

Ввод строки с клавиатуры (в общем виде) :

```
gets channelid ?variable?
```

Считывание с указанием переменной:

```
gets stdin var
puts $var
```

Считывание без указания переменной:

```
set var [gets stdin]
puts $var
```

Выполнение вычислений в Tcl

Синтаксис:

```
expr arg ?arg arg ...?
```

Вычисление значения:

```
expr 123 + 456
```

Вычисление составного значения:

```
set a 2
set b 5
expr 2 * "$a.$b"
```

Математические операции:

```
+, -

*, /, %

**
```

Побитовые операции:

```
~, &, |, ^, >>, <<
```

...

Команда сравнения if elseif else (1)

Синтаксис:

```
if expr1 ?then? body1 elseif expr2 ?then? body2 elseif ... \
?else? ?bodyN?
```

Пример кода:

```
set x 1

if {$x == 2} {puts "$x равно 2"} else {puts "$x не равно 2"}


if {$x != 1} {
        puts {$x != 1 (не равно)}
} else {
        puts {$x равно 1}
}
```

Команда сравнения if elseif else (2)

Пример на if elseif:

```
set x 5

if {$x == 2} {
  puts "$x равно 2"
} elseif {$x == 3} {
    puts "$x не равно 2"
} elseif {$x == 4} {
    puts "$x не равно 2"
} elseif {$x == 5} {
    puts "$x не равно 2"
} elseif {$x == 6} {
    puts "$x не равно 2"
} else {
    puts "Ни одно из рассмотренных"
}
```

Работа со строками в Tcl : сравнение

Синтаксис:

```
string compare ?-nocase? ?-length <число>? string1 string2
string equal   ?-nocase? string1 string 2
```

Пример кода на string compare:

```
set x "Hello"

if {[string compare $x "Hello"] == 0} {
  puts "$x is Hello"
} else {
  puts "$x is not Hello"
}
```

Пример кода на string equal:

```
set x "Hello"

if {[string equal $x "Hello"]} {
  puts "$x is Hello"
} else {
  puts "$x is not Hello"
}
```

Работа со строками в Tcl : обход строки

Получение символа в строке по индексу – команда string index
Синтаксис:

```
string index string_val index_val
```

Получение подстроки по диапазону – string range
Синтаксис:

```
string range string_val index1 index2
```

Получение длины строки – string length
Синтаксис:

```
string length string
```

Команда множественного выбора switch (1)

Синтаксис:

```
switch ?options? string pattern body ?pattern body ...?
```

Пример кода:

```
set x 4

switch $x {
        1 -
        3 -
        5 -
        7 -
        9 { puts "Value is odd" }
        2 -
        4 -
        6 -
        8 { puts "Value is even" }
        default { puts "Value is greater then 10"}
}
```

Команда цикла for

Синтаксис:

```
for start test next body
```

```
for {set x 0} {$x<10} {incr x} {
  puts "x is $x"
}
```

Операторы break, continue:

```
for {set x 0} {$x<10} {incr x} {
    if {$x == 5} {
        continue
    }
    puts "x is $x"
}
```

Команда цикла while

Синтаксис:

```
while test body
```

Пример:

```
set x 1

while {$x < 5} {
    puts "x is $x"
    set x [expr {$x + 1}]
}
```

Сколько раз выведется на экран?

```
set x 0

while "$x < 5" {
  puts "x is $x"
  set x [expr {$x + 1}]
}
```

Списки в Tcl: объявление

Объявление списков (аналог массива):

```
set var4 {1 2 3 4 5}
или
set var4 [list 1 2 3 4 5]
```

```
set item_2 [lindex $var4 2]    <- item_2=3
```

Способы объявления списка:

1. объявлением переменной с указанием элементов;
2. командой создания списка с указанием отдельных элементов;
3. командой разделения строки на элементы

Списки в Tcl: команды работы со списками

Объявление списка:

```
set var4 [list 1 2 3 4 5]
```

Добавление элемента в список:

```
lappend var4 6                    <- var4={1 2 3 4 5 6}
```

Обращение к элементу списка:

```
set item_2 [lindex $var4 2]       <- item_2=3
```

Добавление элемента в список:

```
set var4 [linsert $var4 3 9]      <- var4={1 2 3 9 4 5 6}
```

Замена (обновление значения) элемента в списке:

```
set var4 [lreplace $var4 2 2 7] <- var4={1 2 7 9 4 5 6}
```

Определение вхождения элемента в список:

```
lsearch -exact $var4 2            <- 1
```

Обход списков: команда foreach

Синтаксис команды foreach:

```
foreach varName list body
```

```
#!/usr/bin/tclsh

set var { 1 2 3 4 5 6 }

foreach i $var {
  puts "Item = $i"
}
```

```
#!/usr/bin/tclsh

set var { 1 2 3 4 5 6 }

set j 0
foreach i $var {
  puts "Item no.$j = $i"
  incr j
}
```

Работа с аргументами командной строки в Tcl

За аргументы командной строки в Tcl отвечают три переменные:

1. argc – число передаваемых аргументов;
2. argv0 – имя скрипта;
3. argv – список аргументов.

```
#!/usr/bin/tclsh

puts "Script name         : $argv0"
puts "Args count          : $argc"

set x 0

while {$x < $argc} {
  puts " Arg [expr {$x + 1}]   : [lindex $argv $x]"
  incr x
}
```

Получение информации о файлах

Проверка существования файла
Синтаксис:

```
file exists name
```

Проверка того, является ли файл файлом, каталогом, является ли файл исполняемым
Синтаксис:

```
file isfile name
file isdirectory name
file executable name
```

Извлечение расширения файла
Синтаксис:

```
file extension name
```

Файловый ввод-вывод в Tcl: чтение файлов

```
#!/usr/bin/tclsh

set fp [open "data.txt" r]
set file_data [read $fp]
close $fp

set data [split $file_data "\n"]

foreach line $data {

  puts stdout $line

}
```

Домашнее задание

Для нечётных вариантов

Принять в качестве аргументов командной строки имена файлов. Убедиться, что его имя содержит расширение «.sp». Если нет, скрипт считает, что это не SPICE-нетлист, ругается и выходит.

Для каждого существующего нетлиста нужно вывести статистику:

- есть ли в нетлисте подсхемы (subckt) – вывести «yes» или «no»;
- вывести имена всех подсхем;
- вывести суммарное число строк, НЕ занимаемых подсхемами.

Для чётных вариантов

Принять в качестве аргументов командной строки имена файлов. Убедиться, что его имя содержит расширение «.v». Если нет, скрипт считает, что это не Verilog-нетлист, ругается и выходит.

Считать данные из файла и вывести статистику:

- есть ли в нетлисте модули (module) – вывести «yes» или «no»;
- вывести имена всех модулей (считаем, что между именем и «(» стоит пробел);
- вывести суммарное число строк, занимаемых модулями.

Работаем со списками! Списки имён подсхем, списки имён модулей!