

CS 626 – 636 Spring 2009

From Lecture 1: More FORTH Facts

Languages really are all alike. They wear different clothes, but they are as much alike inside as the photographs on the website. Big, little, pretentious, evolved, young, and old – they have the same parts.

- This language really is free format. You can put the whole program on one line. However, you will never get it debugged. Can is not the same as should.
- How do you read in a number? Well, you CAN, but that is not normal. Normally, you use the system interactively. If you want to add 3 and 4, you type 3 4 + . Newlines don't matter. FORTH is completely free format.
- All dereference in FORTH is explicit. @ When you write xx in a program, it means the address of xx. Most common FORTH error: forgetting the required spaces everywhere. Second most common error: forget the @ when you want the value of a variable.
- The number 1 is a defined symbol in the dictionary. Looking it up is faster than number conversion and you use 1 frequently. (FORTH 83)
- Does FORTH support polymorphism?
Well, that is one of my favorite assignments: Write a polymorphic + function. How? Associate a type tag with every object. (Yes, you can define new types of objects) and override the + function to make it test the type tag before doing anything. Simple. Of course, that is how polymorphism is implemented in C++ and Java.
- Does FORTH support pointers? That is another one of my favorite assignments. No pointer functions are built in. BUT you can allocate memory and get the address of that memory as an integer. You can store that address. You can do address arithmetic. Therefore, you have pointers.
- FORTH can implement recursion.
- In any language, to do either integer-divide or mod, you do a division operation. The only difference is which part gets returned as the answer. For divide, you get the quotient. For mod, you get the remainder. In C, you must do that expensive divide operation twice if you want both answers.
FORTH is different: it has a /mod operation that returns both. This is possible because FORTH uses the parameter stack to return results and can return as many results as wanted. We are freed from the 1-result tyranny. The result is that you have to painstakingly keep track of how many things are on the stack.
- PI is a transcendental number. It is not an integer and not even a real. As I said, all you have in FORTH is ints. You can implement fixed point (mantissa, exponent) or you can do your best with ints. What does this mean in practice? You cannot pre-compute PI. Pi is 22 / 7 or 355 / 113 10 22 * 7 / is 220 / 7 is 31. This answer is good enough. In contrast, 22 7 / 10 * is 30, which is not good enough. You must work 22 / 7 into your formula carefully. If you do all the * before any of the /, you get the best accuracy. You also overflow sooner.

- 10 22 7 */ means multiply first, then use the double-length result in the /. Fact: When you multiply two numbers of roughly the same size, the result is twice as long. (Digits in the product = sum of digits in the multiplier and multiplicand).

So, when you write your formulas for integer arithmetic, you have to compromise between accuracy and overflow. Part of the circle assignment is to choose a formula and find out what the largest input is, before overflow happens in the area. If you make it big before you operate, you have to watch out for overflow.

- Here are the meanings of some of the FORTH operators.
 - 10 5 / is 2 Put the operator between the operands and read left to right.
 - Assume xx is a variable:
 10 xx ! (value address !) (Store 10 in xx.)
 xx @ 3 * xx ! (@ is fetch) (multiply xx by 3)
 - ?dup means "Duplicate if non-zero". It is very important: you need it when you write a while loop. (A look at the gcd code will tell you why).
 - 2dup Duplicate the 2 numbers on top of the stack. This is defined because it is so very common that you want to duplicate a pair of parameters.
 - swap : Swap the 2 numbers on top of the stack.
 - . removes a value from the stack and kicks it onto the screen.
 - 65 emit (display the ASCII value 65 to the screen.) (you see A)
 - cr (inserts a carriage return in the output.)
 - < means to compare the top two things on the stack and answer true or false.
 - if looks at the top of the stack and interprets it as true or false.
 - 0 is false, anything else is true, just like C.

- Let us write some functions.

```

: hello ." Hello World" ; ok
: four 2 2 + ." the answer is " . ; ok
: fun ( n1 n2 >>> n3 )                compiled
  2dup                                ( n1 n2 n1 n2 ) compiled
  swap                                ( n1 n2 n2 n1 ) compiled
  ." The sum of " . ( n1 n2 n2 )      compiled
  ." and " . ( n1 n2 )                compiled
  ." =" + ( n3 )                      compiled
  dup                                  ( n3 n3 )      compiled
  .                                    ( n3 )        compiled
; ok

```

- OK is the system prompt. Now we will call the functions.

```

hello Hello World ok
hellp
:5: Undefined word
>>>hellp<<<
Backtrace:

```

```

$304EB4 throw
$311620 no.extensions
$30502C interpreter-notfound1
ok
10 99 fun The sum of 10 and 99 =109 ok
four the answer is 4 ok
bye

```

- Interpretation of the call on four:
 - put 2 on the stack.
 - put 2 on the stack again.
 - add the top two things on the stack.
 - print a message.
 - print the value at the top of the stack
- The code for gcd is on the website and is repeated, below, followed by some test data. This code is not the only way to use stack operations to compute gcd. Two people will do it in two different ways. No magic. No inspiration needed. Just figure out how many copies you need of each number and copy it while it is nearby.

```

\ Greatest Common Divisor Function ( n1 n2 >> gcd )
: gcd
  2dup < if      ( smaller arg should be at stack top. )
  swap then     ( larger_arg smaller_arg      )
  begin         ( lg sm                        )
    ?dup
  while         ( exit on 0 remainder, answer is on stack. )
    swap over   ( sm lg sm                      )
    mod         ( sm remainder )
  repeat       ( ready for next iteration )
;             ( return from subroutine. )

```

```

CR
3 15 gcd . CR
2 15 gcd . CR
6 45 gcd . CR

```

- Let us step through the gcd code, one operation at a time. Call: 3 21 gcd
Each line shows an operation and the contents of the stack.

```

2dup      3 21 3 21
<        3 21 true  now the if sees the true
if       3 21      (if removes the true and does the swap.)
swap     21 3
begin    ----- (remember this place)
?dup     21 3 3    (it was not 0 so we duped it.)
while    (sees the 3 and removes it. 3 is true, so the loop goes on.)
         21 3

```

```

swap      3 21
over      3 21 3
mod       3 0 (21 mod 3 is 0)
repeat    ----- go back to begin.
?dup      3 0 ( does nothing because top of stack is 0 )
while     (sees the 0, removes it, leaves the loop.)
          3 (The correct answer is left on the stack.)
          (control goes to the line after repeat.)

```

- Lets do another (harder) example 21 99 gcd (21 99 >>> n)

```

2dup      ( 21 99 21 99 )
<         ( 21 99 true )
if        ( 21 99 ) do the controlled clause.
swap      ( 99 21 )
then      ----- marks the end of the if clause.
begin     ----- remember the top of the loop.
?dup      ( 99 21 21 )
while     ( 99 21 ) and stay in the loop because 21 is true.
swap      ( 21 99 )
over      ( 21 99 21 )
mod       ( 21 15 )
repeat    ----- sends you back to the begin.
?dup      ( 21 15 15 )
while     ( 21 15 ) and stay in the loop
swap      ( 15 21 )
over      ( 15 21 15 )
mod       ( 15 6 )
repeat    ----- sends you back to the begin.
?dup      ( 15 6 6 )
while     ( 15 6 ) and stay in loop
swap      ( 6 15 )
over      ( 6 15 6 )
mod       ( 6 3 )
repeat    ----- back to begin
?dup      ( 6 3 3 )
while     ( 6 3 ) and stay in loop
swap      ( 3 6 )
over      ( 3 6 3 )
mod       ( 3 0 )
repeat    ----- back to begin
?dup      ( 3 0 )
while     ( 3 ) and leave the loop.
;

```

Second week's assignment. Standard circle program in FORTH. Given a diameter, print out radius, circumference, area. Make sure circles with radii 10 and 11 have different circumferences. (Think)