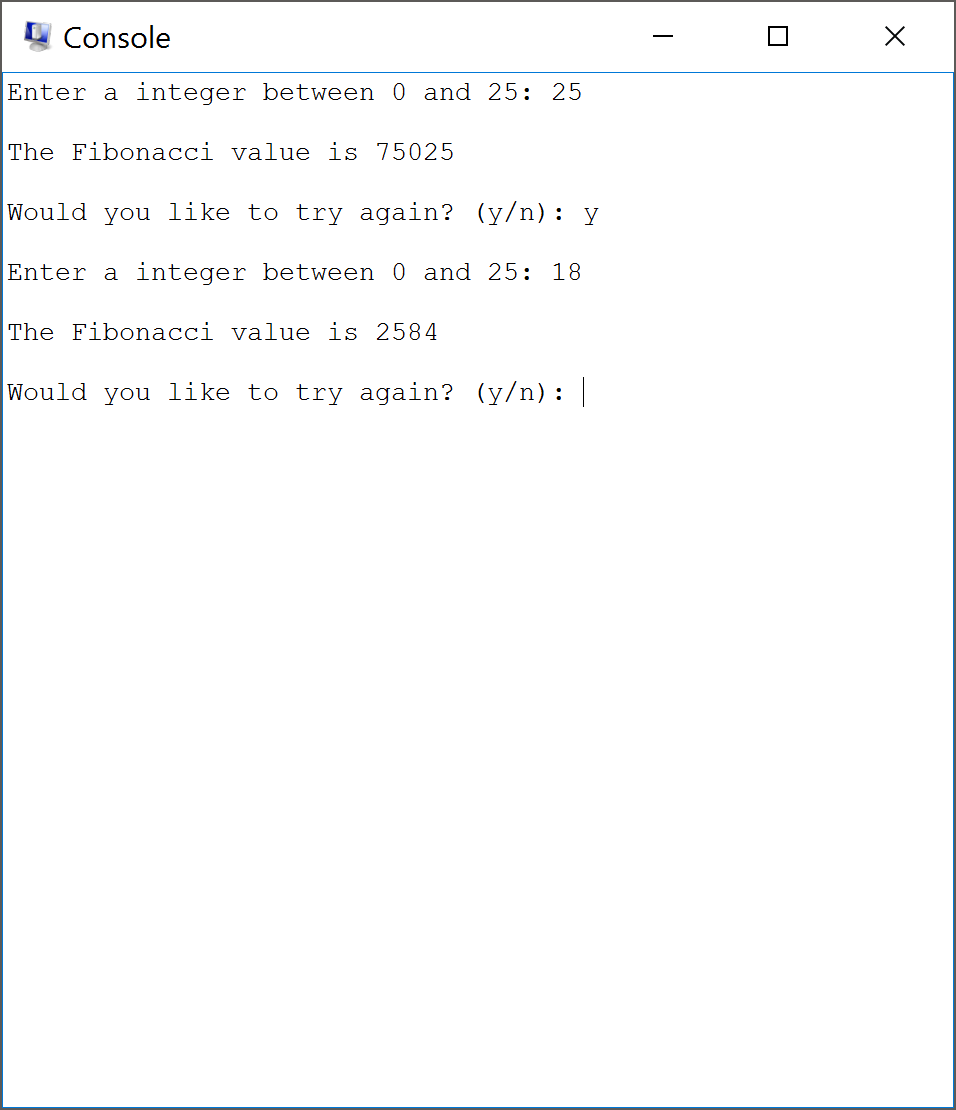
Joshua Pollock

CS 200 Homework 9

19 April 2017

Example Output:



Code (Will also be uploaded to BBLearn):

# TITLE fibonacci (HW9.s)

# This program handles the I/O for a fibonacci function.

.data

# variables

IntPrompt: .asciiz "Enter a integer between 0 and 25: "

OutStr: .asciiz "\nThe Fibonacci value is "

LowError: .asciiz "\nYour input was too small. Try again: "

HighError: .asciiz "\nYour input was too large. Try again: "

AgainStr: .asciiz "\nWould you like to try again? (y/n): "

NewLine: .asciiz "\n"

YesNoBuf: .space 5 # Plenty of room for 'yes' or 'no'

IntIn: .word 0

IntMin: .word 0

IntMax: .word 25

.text

.globl main

again: # print a newline on subsequent returns to main

la $a0, NewLine # point to NewLine

li $v0, 4 # print\_string

syscall

main: # start of the main procedure

move $t9,$s1 #preserving $s1 (Probably not needed but good practice)

move $s1,$ra # Saving original $ra

# Get an integer

la $a0, IntPrompt # point to IntPrompt

li $v0, 4 # print\_string

syscall

GetInt:

li $v0, 5 # read\_integer

syscall

move $t0, $v0 # move input before it gets changed

# check if below min

lw $a1, IntMin # load our lower bound

bge $v0, $a1, BigEnough # if good, try next check

la $a0, LowError # point to Error string

li $v0, 4 # print\_string

syscall

j GetInt

# check if above max

BigEnough:

lw $a1, IntMax # load our upper bound

ble $v0, $a1, SmallEnough # if good, try next check

la $a0, HighError # point to Error string

li $v0, 4 # print\_string

syscall

j GetInt

SmallEnough:

# save the input, just in case

sw $v0, IntIn

# Print the text to go with the output

la $a0, OutStr # point to OutStr

li $v0, 4 # print\_string

syscall

# Calls Fib and gets Fibonacci number

######################################################################

move $a0,$t0 #Moves n back into $a0

jal fib # Jump and link to fib

move $a0,$v0 # Moves fibs return into $a0

li $v0, 1 # print\_integer

syscall

######################################################################

# Print a newline before continuing

la $a0, NewLine # point to NewLine

li $v0, 4 # print\_string

syscall

# Prompt to see if the user wants to do it again

la $a0, AgainStr # point to AgainStr

li $v0, 4 # print\_string

syscall

# Get the input

la $a0, YesNoBuf # point to YesNoBuf

li $a1, 5 # length of buffer

li $v0, 8 # read\_string

syscall

lb $t0, YesNoBuf # load the first character into $t0

# Test if first character is 'Y'

li $t1, 89 # ASCII for 'Y'

beq $t0, $t1, again # equal, so run program again

# Test if first character is 'y'

li $t1, 121 # ASCII for 'y'

beq $t0, $t1, again # equal, so run program again

# Not 'yes', so assume 'no' and end program

j end

# Checks if n<1 and then eithier jumps to the recursive or returns n

fib:

bgt $a0,1,fib\_recursive # If n<1

move $v0,$a0 # else return=n

jr $ra # return to caller

# If the number is recursive go into here and calculate the fibonacci number

fib\_recursive:

addi $sp,$sp,-12 # Adds 3 spaces to the stack

sw $ra,0($sp) # Saves $ra in stack

sw $a0,4($sp) # Saves n to the stack

addi $a0,$a0,-1 # Does n-1

jal fib

lw $a0,4($sp) # puts n into $a0

sw $v0,8($sp) # Saves n-1 return into the stack

addi $a0,$a0,-2 # Does n-2

jal fib # Jumps back to fib

lw $t0,8($sp) # Puts return value of n-1 into $t0

add $v0,$t0,$v0 # Adds (n-1) and (n-2)

lw $ra, 0($sp) # puts $ra back into $ra

addi $sp,$sp,12 # Kills the stack

jr $ra # Return to caller

# Restores $ra and ends program

end:

move $ra,$s1

move $s1,$t9

jr $ra