Dylan Johnson

CSC 330

Language Design and Implementation

Week 2

Lesson 1:

logical function isPrime(num)

implicit none

integer :: i, num ! Declares variables

if (num < 2) then ! no "math" needed if number is < 2

isPrime = .false.

else

isPrime = .true.

do i = 2, num - 1

if (mod(num, i) == 0) then ! checks if the number is divisable by more then just itself and 1

isPrime = .false.

return

end if

end do

end if

return

end function isPrime

logical function isMersennePrime(primeNum)

implicit none

! Declares variables

integer :: primeNum, checkMPrime

logical :: isPrime

checkMPrime = (2 \*\* primeNum) - 1 ! stores the potential mersenne prime number

isMersennePrime = isPrime(checkMPrime) ! checks if the prime number is a mersenne prime

return

end function isMersennePrime

Program main

implicit none

! Declares variables

integer :: start = 2, finish = 0

logical :: isNumberPrime, isNumberMersenne, isPrime, isMersennePrime

do while (finish /= 8) ! stops after the first 8 have been found

isNumberPrime = isPrime(start) ! starting with 2, checks if a number is prime

if (isNumberPrime .eqv. .true.) then ! if it is prime, moves onto check if it is a mersenne prime

isNumberMersenne = isMersennePrime(start)

if (isNumberMersenne .eqv. .true.) then

print\*, (2 \*\* start) - 1 ! prints the number if it is a mersenne prime

finish = finish + 1

end if

end if

start = start + 1

end do

End Program main

Lesson 2:

#include <stdio.h>

#define ROWS 5

#define COLS 5

#define FILENAME 100

void multiplyMatrix(int matrix[ROWS][COLS], int scalar) {

int result[ROWS][COLS]; // Declares results matrix

for (int i = 0; i < ROWS; i++) { // Loops through the rows

for (int j = 0; j < COLS; j++) { // Loops through the Columns

result[i][j] = matrix[i][j] \* scalar; // performs the math per scaler variable

}

}

// Prints the results of the above scaling

printf("Results:\n");

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLS; j++) {

printf("%d ", result[i][j]); // Prints the results formatted like the matrix sample input file

}

printf("\n"); // makes sure only 5 to a line

}

}

int main() {

FILE \*file;

int matrix[ROWS][COLS];

int scalar;

char filename[FILENAME];

printf("What file would you like to load a 5x5 matrix from? ");

scanf("%s", filename); // Looks for given file name

// If the given file name doesn't exist, program errors out

file = fopen(filename, "r");

if (file == NULL) {

printf("Error opening the file.\n");

return 1;

}

// If the file does exist, it reads the input as teh 5x5 matrix and

// stores it int he matrix integer 2d array

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLS; j++) {

fscanf(file, "%d", &matrix[i][j]);

}

}

fclose(file);

// Gets the amount it should be multiplied by

printf("Enter the scaler value: ");

scanf("%d", &scalar);

// Performs the multiplication in above function

multiplyMatrix(matrix, scalar);

return 0;

}