Alejandro Martinez
Programming 3
Program #2 Babylonian Calculations
Pseudocode

Display program purpose: This program determines the 'perfect numbers' from 1-100,000, including both 1 and 100,000, and then computes the square root of each perfect number using the Babylonian approach.

Display results: Using explainTheProgram(), requestInput(), numberRange (int numberToCheck), perfectNumberCalculation(int validatedUserInput), isTheNumberPerfect(int numberToCheck, int userInput), findFactors(int perfectNumber), expectedSquareRoot (int perfectNumber), generateGuess (int perfectNumber), babylonianCalculation(int perfectNumber, int userInput, int guess), the program will display all the perfect numbers from 1- 100,000 and will compute the Babylonian approximation to calculate the square root of that perfect number.

Main(void)

Call the function explainTheProgram() to explain the purpose of the program to the user;

Call function requestInput() and assign it to variable input;

Call the function perfectNumberCalculation(int validatedUserInput) and pass the variable input as the parameter.

Function – void explainTheProgram() This function explains the purpose of the program to the user

Print to the user a welcome statement explaining what this program will do;

Function – int requestInput() Validate the input of the user. This function validates for characters that are not integers or integers that are not in the desired range of accuracy.

Initialize the input variable to 0. The input variable will eventually store the user input;

Initialize the status variable to 0. This variable will change according if the input was validated or not;

Initialize the validatedNumber variable to 0. This variable will check if the number is in the desired range.

Initialize the validatedInput variable to 0. This variable will change according to the status of the validated number.

Initialize a new variable called radius to 0 as a float;

Do

Initialize the variable checkChar to -1;

Print to the user the number range he is suppose to type.;

Assign variable status to Scanf(input);

While checkChar is not '\n'

Assign checkChar to getChar();

If checkChar is not a digit using isDigit(checkChar) and checkChar is not '\n'

Status = 0;

If the Status is equal to 1

validatedNumber is equal to the input;

call function numberRange(validatedNumber) and assign it to validatedInput;

While

status is not equal to 1 or validatedInput is not equal to 0;

Return validatedNumber;

Function – int numberRange(int numberToCheck) - This function validates if the input number of the user is within the required range.

Initialize the statusOfNumber variable to 1 as an int;

If the numberToCheck is either lower than the LOWER_BOUND_FOR_INPUT or higher than the UPPER_BOUND_FOR_INPUT

Print to the user to input the number again in the rquired range

statusOfNumber equals to 0

Return statusOfNumber;

Function – perfectNumberCalculation(int validatedUserInput). This function will iterate for our targeted range [1-100,0000]. It will call function isTheNumberPerfect(counter, userInput) to know if the number in the range is a perfect number.

Initialize the variable counter to 0;

for(counter = LOWER_BOUND_FOR_PERFECTNUMBERS;
counter <= UPPER_BOUND_FOR_PERFECTNUMBERS; counter++)

compute function isTheNumberPerfect (counter, validatedUserInput);

Function – void isTheNumberPerfect(int numberToCheck, int userInput). This function will add the factors of the parameter 'numberToCheck' and will state if the number is perfect or not. This will be done by adding the factors and seeing if adds to the 'numberToCheck'

Initialize the variables counter and sum to 0;

for (counter = 1; counter < numberToCheck; counter++)

sum is equal to sum plus counter;

if the sum equals the numberToCheck

print the perfect number;

call the function findFactors();

initialize a guess;

call the function generateGuess(numberToCheck) and assign it to the guess;

call the function expectedSquareRoot (numberToCheck);

call the function babylonianCalculation(numberToCheck, userInput, guess);

Function – void findFactors (int perfectNumber) - This function will print the factors of the perfect number.

Initialize counter, factor and sum to 0;

for (counter = 1; counter < perfectNumber; counter++)

if the mod between the perfectNumber and the counter is 0

sum = sum + counter;

make the factor equal to the counter and print the factor

Function – void expectedSquareRoot(int perfectNumber) - This function will use the square root from the math library to know the exact square root of the perfect number.

Initialize expectedSQRoot to a float;

expectedSQRoot equals to the sqrtf(expected square root);

print the variable expectedSQRoot;

Function – void babylonianCalculation(int perfectNumber, int userInput, int guess) - The purpose of this function is to calculate the approximated square root using the babylonian method. As explained in the documentation this calculation will use the userInput to have an accurate result.

Initialize desiredSqrt to a float and make it equal to perfectNumber;

Initialize the computedGuess to a float and make it equal to the guess.

Initialize the accuracy and make it equal to 1 / (pow(10, userInput())) as a float;

Initialize a temp variable to 0 as a float;

Initialize the delta to 0 as a float;

Initialize the numberOfIterations to 0 as an int;

While fabs(temp – computatedGuess is greater than the accuracy)

temp = computatedGuess;

computatedGuess = ((computatedGuess + (desiredSqrt/computatedGuess)) / 2;

delta = fabsf(temp - computatedGuess);

numberOfIterations++;)

print the variable perfectNumber;

print the numberOfIterations;

print the delta;

print the userInput;

Function – float generateGuess (int perfectNumber) - This function will guess estimate the initial guess required by the babylonian method..

Initialize numberToApproximate to 0 as a float;

numberToApproximate equals to the perfectNumber / 2

print the variable numberToApproximate;

Return numberToApproximate;
