OSSabado6AM PGM3

CABARCAS PERDOMO ALEJANDRO

(20191176712)

Tabla de contenido:

* Planteamiento programa
* Diagrama de pasada
* Diagrama de uso
* Diccionario de datos
* Código incrustado en wordpad
* Demostración grafica del proceso de ejecución
* Conclusiones

PLANTEAMIENTO DEL PROGRAMA:

PGM3. Leer 10 números enteros, almacenados en un vector y determinar cuántos números terminan en digito primo.

DIAGRAMA DE PASADA:

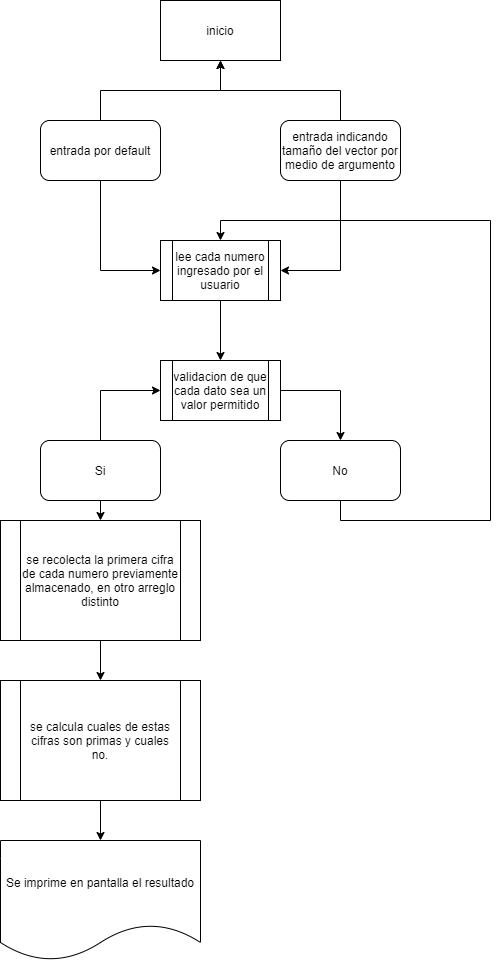
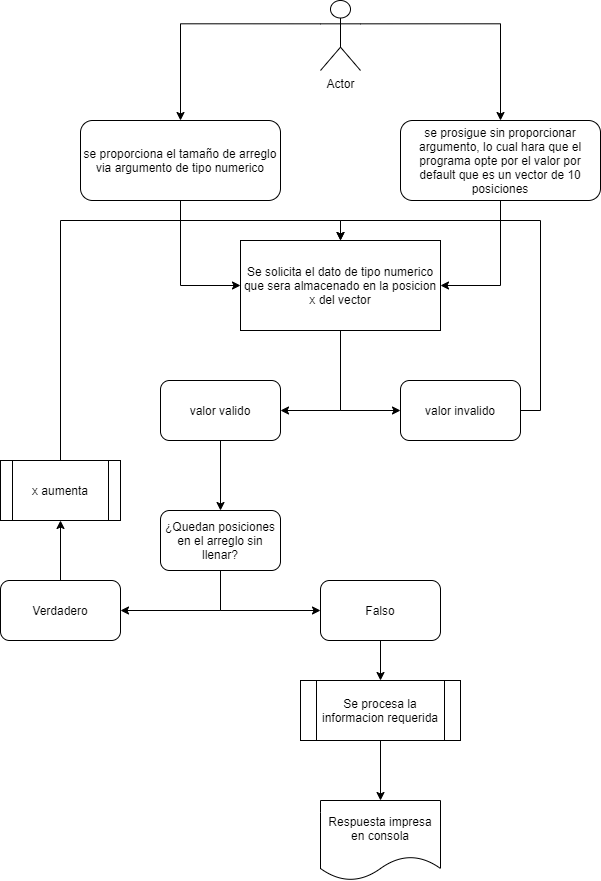


DIAGRAMA DE USO:



DICCIONARIO DE DATOS:

Arreglos:

numeros[]; int

modulos[]; int

Variables:

limite; int

prueba; int

numeroIngresado; int

nozero; int

prueba; int

len; int

count; int

primo; int

numero; int

modulo; int

#!/bin/bash

numeros=()

modulos=()

#se recibe un argumento el cual determina la cantidad de numeros primos a analizar

limite=$1

if [[ $limite -eq 0 ]]; then

#si no se ingresa ningun argumento, o uno invalido

#se asigna la longitud del arreglo por defecto

limite=10

fi

echo "---------------------------------------------"

echo "-Bienvenido al comprobador de numeros primos-"

echo "---------------------------------------------"

echo ""

echo "------------------------------------------------------"

echo "- Debe digitar 10 numeros que se almacenaran en      -"

echo "- un arreglo que posteriormente sera evaluado        -"

echo "- para devolver como salida cuales de estos numeros  -"

echo "- poseen en su unidad un numero primo.               -"

echo "------------------------------------------------------"

echo ""

#le pido datos al usuario por medio de una ciclo, para asi llenar los datos

for((i=0; i < $limite; i++))

do

#la variable prueba sirve para validar lo ingresado por el usuario fue un numero

prueba=0

echo "digite el numero a almacenar en la posicion " $i

read numeroIngresado

#nozero almacena el numero ingresado descartando los ceros a la izquierda si los tiene

nozero=$(echo $numeroIngresado | sed 's/^0\*//')

echo ""

#este condicional se encarga de evaluar que el numero ingresado no sea negativo

if [[ $numeroIngresado -ge 0 ]]; then

  let prueba=nozero/nozero

fi

#valido que el dato ingresado sea un numero antes de incluirlo en el arreglo

if [[ $prueba == 1 ]]; then

  #se usa la variable prueba nuevamente para que en caso de que la persona halla

  #ingresado un numero decimal, lo transforme en uno entero

  let prueba=nozero\*1

  numeros[i]=$prueba

else

  echo ""

  echo "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

  echo "x Por favor ingrese un valor valido x"

  echo "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

  echo ""

  let i=i-1

fi

done

#creo una variable que contiene la longitud del arreglo

len=${#numeros[@]}

#recorre el vector numeros para asi llenar el vector modulos con la primera cifra de cada numero

#almacenado en la misma posicion del vector numeros

for ((i=0; i < len;i++))

do

let modulos[i]=${numeros[i]}%10

done

count=0

#con la ayuda de un ciclo recorro cada elemento del arreglo

for i in ${numeros[@]}

do

#j se encarga de servir de indice para el while anidado

j=2

#todo numero empieza siendo tomado como primo, hasta que se valida

primo=1

numero=${modulos[count]}

#se toma el numero en la posicion del arreglo, y se recorren

#todos los numeros anteriores a este operando cada vuelta

while [ $j -lt ${modulos[count]} ]

do

numero=${modulos[count]}

#se valida el resultado si el numero es 1

if [ ${modulos[count]} -eq 1 ]

then

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

echo "el numero 1 no es primo"

else

#echo "contador" $count

#echo "valor arreglo" ${numeros[count]}

#echo "valor de i" $i

#echo "valor de j" $j

#se halla el modulo de todos y cada uno de los numeros anteriores

#al numero alojado en esa posicion del arreglo, si algun numero de estos

#es divisible entre el numero que aloja esa posicion del arreglo, osea,

#su modulo es 0, la variable primo cambia de 1 a 0, lo cual significa

#que ese numero de esa posicion del arreglo no es primo

let modulo=${modulos[count]}%$j

#echo "modulo" $modulo

fi

if [ $modulo -eq 0 ]

then

primo=0

#una vez se sabe que el numero no es primo, se fuerza la salida del ciclo

#puesto que no es necesario seguirlo recorriendo

j=${modulos[count]}

fi

#se hace el incremento de la variable j

let j=j+1

done

#se evalua el valor de la variable primo que determina si el numero es o no primo

if [ $primo -eq 0 ]

then

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

echo "La ultima cifra del numero, " $i " NO es prima"

else

if [ $numero -eq 0 ]

then

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

echo "La ultima cifra del numero, " $i " NO es prima"

else

if [ $numero -eq 1 ]

then

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

echo "La ultima cifra del numero, " $i " NO es prima"

else

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

echo "La ultima cifra del numero, " $i " SI es prima"

fi

fi

fi

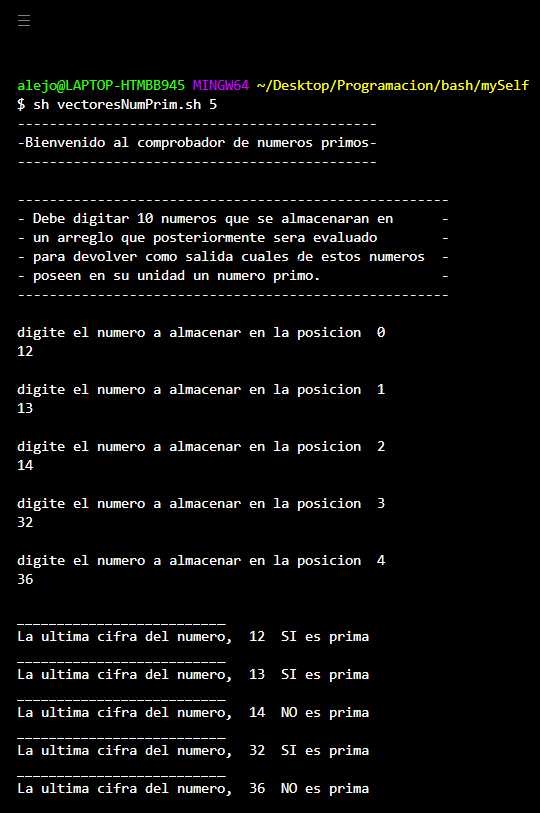
#se hace el incremento de la variable count

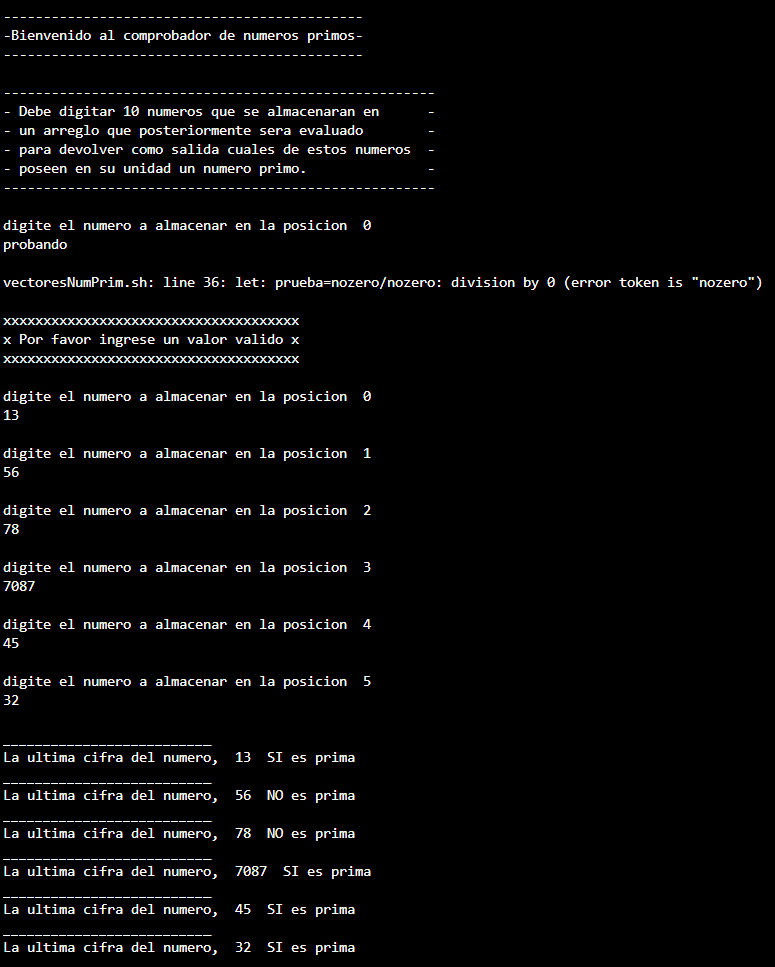
let count=count+1

done

por si el anterior código no se ve de forma concisa, también lo adjunto usando wordpad.

Demostración grafica del proceso de ejecución:





Conclusiones:

El programa se ejecuta de forma satisfactoria y complaciente a los requisitos estipulados en referencia al problema previamente planteado.

Cuenta tanto con su código que se encarga de validar que el usuario no pueda terminar de forma abrupta el programa por un valor invalido, así como el procesamiento correcto de las solicitudes si todos los datos requeridos fueron ingresados satisfactoriamente.

El ultimo punto a destacar es que le permite al usuario por medio de un argumento justo antes de proceder la ejecución, establecer el tamaño del vector que se llenara con los datos que el proporcionara.