Operating Systems

Lab 6 -Assignment

1. Shell script to implement all basic array operations.

2. Shell script to accept and print array elements.

3. Shell script to implement bubble sort and binary search.

Shell script to implement all basic array operations.

#!/bin/bash

array1[0]=one

array1[1]=1

echo ${array1[0]}

echo ${array1[1]}

array2=( one two three )

echo ${array2[0]}

echo ${array2[2]}

array3=( [9]=nine [11]=11 )

echo ${array3[9]}

echo ${array3[11]}

read -a array4

for i in &quot;${array4[@]}&quot;

do

echo $i

done

exit 0

OUTPUT



Various Operations on Arrays

Many of the standard string operations work on arrays . Look at the following sample script which implements some operations on arrays (including string operations).

#!/bin/bash

array=( apple bat cat dog elephant frog )

#print first element

echo ${array[0]}

echo ${array:0}

#display all elements

echo ${array[@]}

echo ${array[@]:0}

#display all elements except first one

echo ${array[@]:1}

#display elements in a range

echo ${array[@]:1:4}

#length of first element

echo ${#array[0]}

echo ${#array}

#number of elements

echo ${#array[\*]}

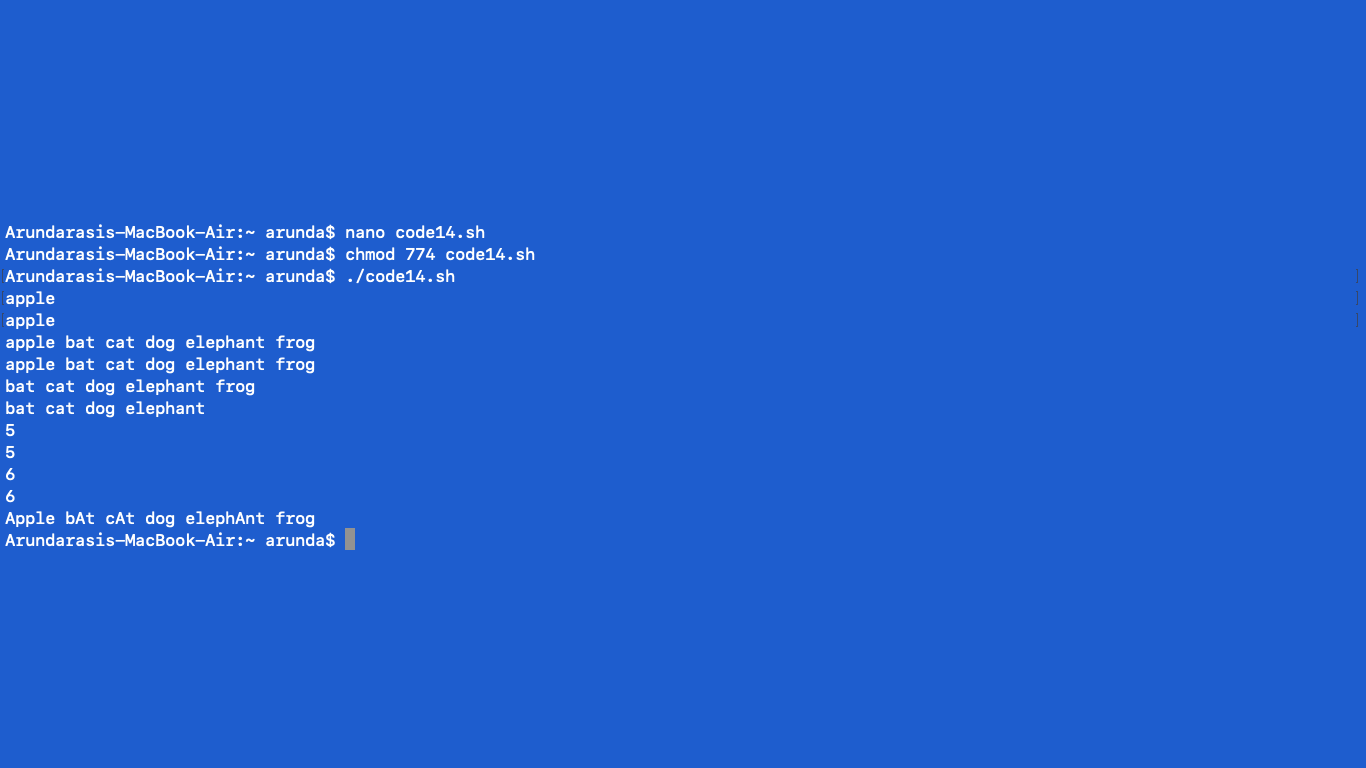
echo ${#array[@]}

#replacing substring

echo ${array[@]//a/A}

exit 0

OUTPUT



Shell script to accept and print array elements.

By Using while-loop

# !/bin/bash

# To declare static Array

arr=(1 12 31 4 5)

i=0

# Loop upto size of array

# starting from index, i=0

while [ $i -lt ${#arr[@]} ]

do

# To print index, ith

# element

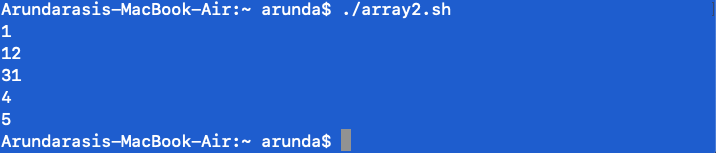
echo ${arr[$i]}

# Increment the i = i + 1

i=`expr $i + 1`

Done

OUTPUT



By Using for-loop

# !/bin/bash

# To declare static Array

arr=(1 2 3 4 5)

# loops iterate through a

# set of values until the

# list (arr) is exhausted

for i in &quot;${arr[@]}&quot;

do

# access each element

# as $i

echo $i

done

OUTPUT



To Read the array elements at run time and then Print the Array.

1. Using While-loop

# !/bin/bash

# To input array at run

# time by using while-loop

# echo -n is used to print

# message without new line

echo -n &quot;Enter the Total numbers :&quot;

read n

echo &quot;Enter numbers :&quot;

i=0

# Read upto the size of

# given array starting from

# index, i=0

while [ $i -lt $n ]

do

# To input from user

read a[$i]

# Increment the i = i + 1

i=`expr $i + 1`

done

# To print array values

# starting from index, i=0

echo &quot;Output :&quot;

i=0

while [ $i -lt $n ]

do

echo ${a[$i]}

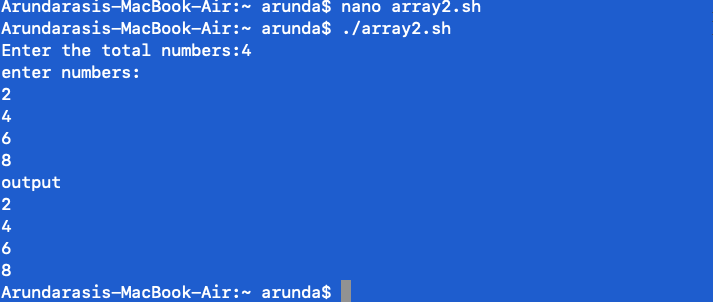
# To increment index

# by 1, i=i+1

i=`expr $i + 1`

done

OUTPUT



Sorting an array in Bash using Bubble sort

# Sorting the array in Bash

# using Bubble sort

# Static input of Array

arr = (10 8 20 100 12)

echo &quot;Array in original order&quot;

echo ${arr[\*]}

# Performing Bubble sort

for ((i = 0; i&lt;5; i++))

do

for((j = i; j&lt;5-i-1; j++))

do

if ((${arr[j]} &gt; ${arr[$((j+1))]}))

then

# swap

temp = ${arr[$j]}

arr[$j] = ${arr[$((j+1))]}

arr[$((j+1))] = $temp

fi

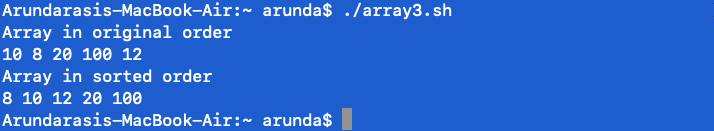
done

done

echo &quot;Array in sorted order :&quot;

echo ${arr[\*]}

OUTPUT



Script to implement bubble sort and binary search

echo Enter array limit

read limit

echo Enter elements

n=1

while [ $n -le $limit ]

do

read num

eval arr$n=$num

n=\`expr $n + 1\`

done

echo Enter key element

read key

low=1

high=$n

found=0

while [ $found -eq 0 -a $high -gt $low ]

do

mid=\`expr \\( $low + $high \\) / 2\`

eval t=\\$arr$mid

if [ $key -eq $t ]

then

found=1

elif [ $key -lt $t ]

then

high=\`expr $mid - 1\`

else

low=\`expr $mid + 1\`

fi

done

if [ $found -eq 0 ]

then

echo Unsuccessful search

else

echo Successful search

fi

OUTPUT

