Kabuk Programlama
Shell Scripting(bash)

Shell Türleri

- Bourne
- Bash
- Z-dhell
- C-shell
- TC-shell
- Korn

- «man bash» yazarsak bilgi ediniriz.
- «which bash» yazarsak shell dosyas n n bulundu u yeri gösterir.

Hangi Kabuktay z?

- echo $SHELL yazarsak
- /bin/sh ise Bourne
- /bin/ksh93 ise Korn
- /bin/bash ise Bash
- /bin/zsh ise Z shell
- /bin/csh ise C shell
- /bin/tcsh ise TC shell

Script ilk sat rlar

- #!/bin/csh #This is a sample C-shell script

  ```
  echo -n the date of today is’ ’        # -n omits new line
  date
  ```

- #!/bin/ksh #This is a sample K-shell script

  ```
  echo “the date of today is \c”        # \c omits new line
  Date
  ```

- #!/bin/bash #This is a sample BASH script

  ```
  echo -n “the date of today is ”      # -n omits new line
  date
  ```

Basit Script(Betik) Nas l Yaz l r?

- "Hello World" script ile baslayal m Uzant s .sh olacak bir dosya içine script yaz l r. Dosyaya,
- #!/bin/bash
- echo "Hello, World " yaz l r
- Çal t rma modu de i tirilir :

chmod a+x /çal lanklasör/hello_world.sh veya

chmod 700 /çal lanklasör/hello_world.sh

- çal lanklasör/hello_world.sh yazarak cal st r.
- pwd çal lan klasöru gosterir.
- Script dosyas na, «komut sat r ndan girilen komutlar + döngü ve karar kontrol yap lar » yaz l r.

De   i   kenler

- De   i   ken tan mlan rken türü belirtmeye gerek yoktur. Ama istenirse declare kelimesiyle de tan mlanabilir. De   i   ken de   erine ula   mak için "$" kullan l r, atama yaparken "$" kullan lmaz. Atamalarda e   it i   aretinin çevresine bo   luk konulmaz.
-
  ```
  #!/bin/bash
  STRING="HELLO WORLD!!!"
  echo $STRING
   declare -i x
   x=10
   x=x+1
   echo $x
  ```

Kelimeler -Stringler

- Atamalarda " kullan rsak de   i   kenlerin de   erini yazar
- Atamalarda ‘ kullan rsak her   eyi text olarak al r:

```
#!/bin/bash
var=2
echo “Merhaba $var”
echo ‘Merhaba $var’
```

Yönlendirme Karakterleri

- ls > dosya.txt :dosyaya yazmak
- sort < dosya.txt > sirali.txt :dosyay s ralay p yazmak
- date >> sirali.txt :dosyan n sonuna ekleme

Klavyeden De er Girme(read)

```
• #!/bin/bash
   clear
  echo " Ad n z  Giri"n
  read name
  echo " Yas n z  giri"n
  read age
  echo " Cinsiyet girin: K/E"
  read sex
  echo " Siz $age yas nda$sex cinsiyetinde
  $name adl  kisisiniz"
```

Örnek

- Ayn s   echo olmadan read komutu ile:

clear

read -p "Ad n z   girin" name

read -p "Yas n z   girin" age

read -p "Cinsiyet girin" sex

echo "Siz $age yas nda $sex $name isimli ki   isiniz"

- Temiz bosluklar basacak sekilde:
- clear
- read -p "Ad n z   girin: " name echo ""
- read -p "Yas n z   girin" age echo ""
- read -p "Cinsiyet girin" sex echo ""
- echo "Siz $age yas nda $sex $name isimli ki   isiniz"

Örnek-2

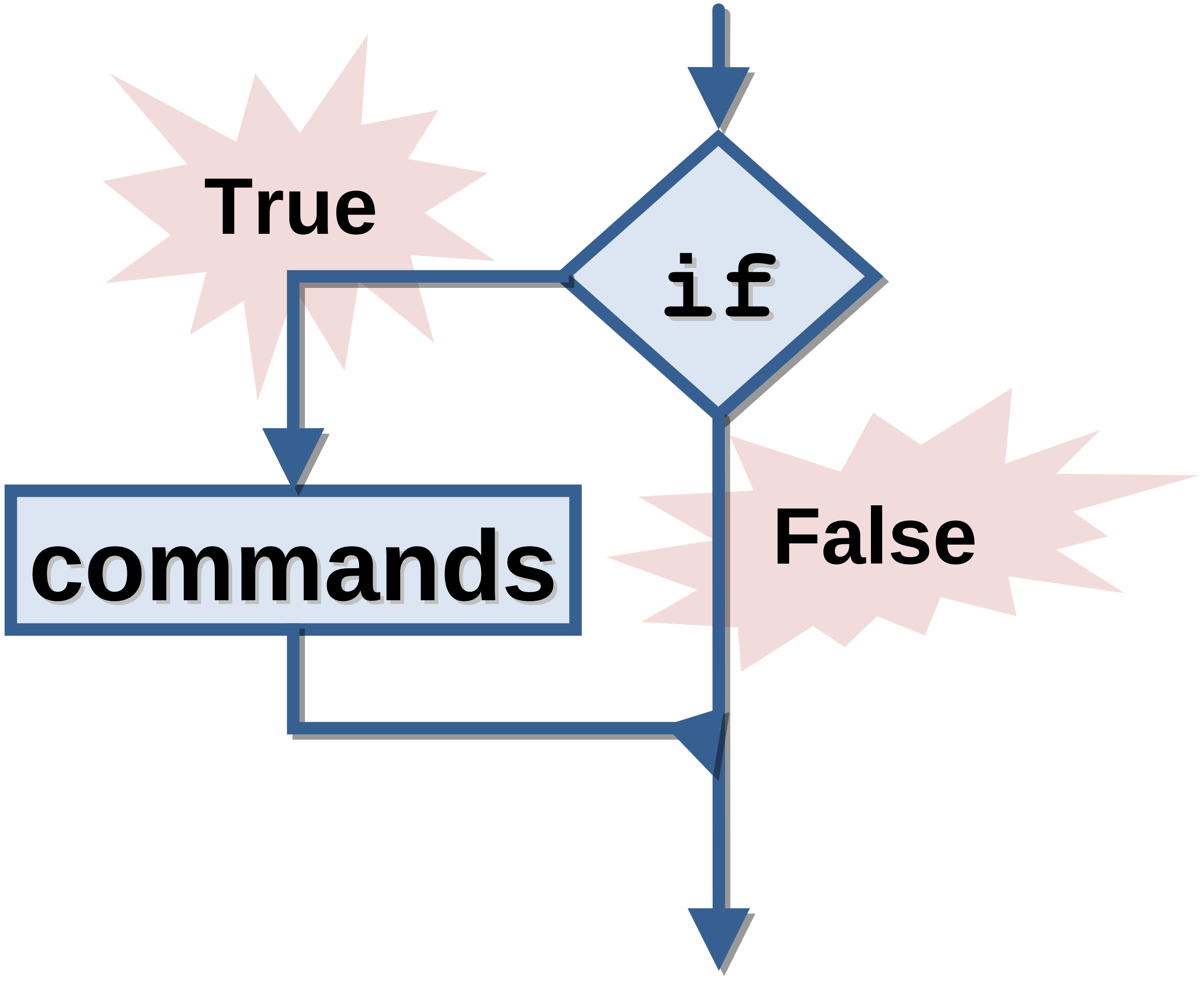
- #!/bin/bash
- echo -n "ilk say y   girin▹ "
- read a
- echo -n "ikinci say y   girin▹ "
- read b
- echo "Aritmetik i   lem▹ "
- sum=$(($a + $b))
- echo "a ve b toplam $sum"

if karar yap lar

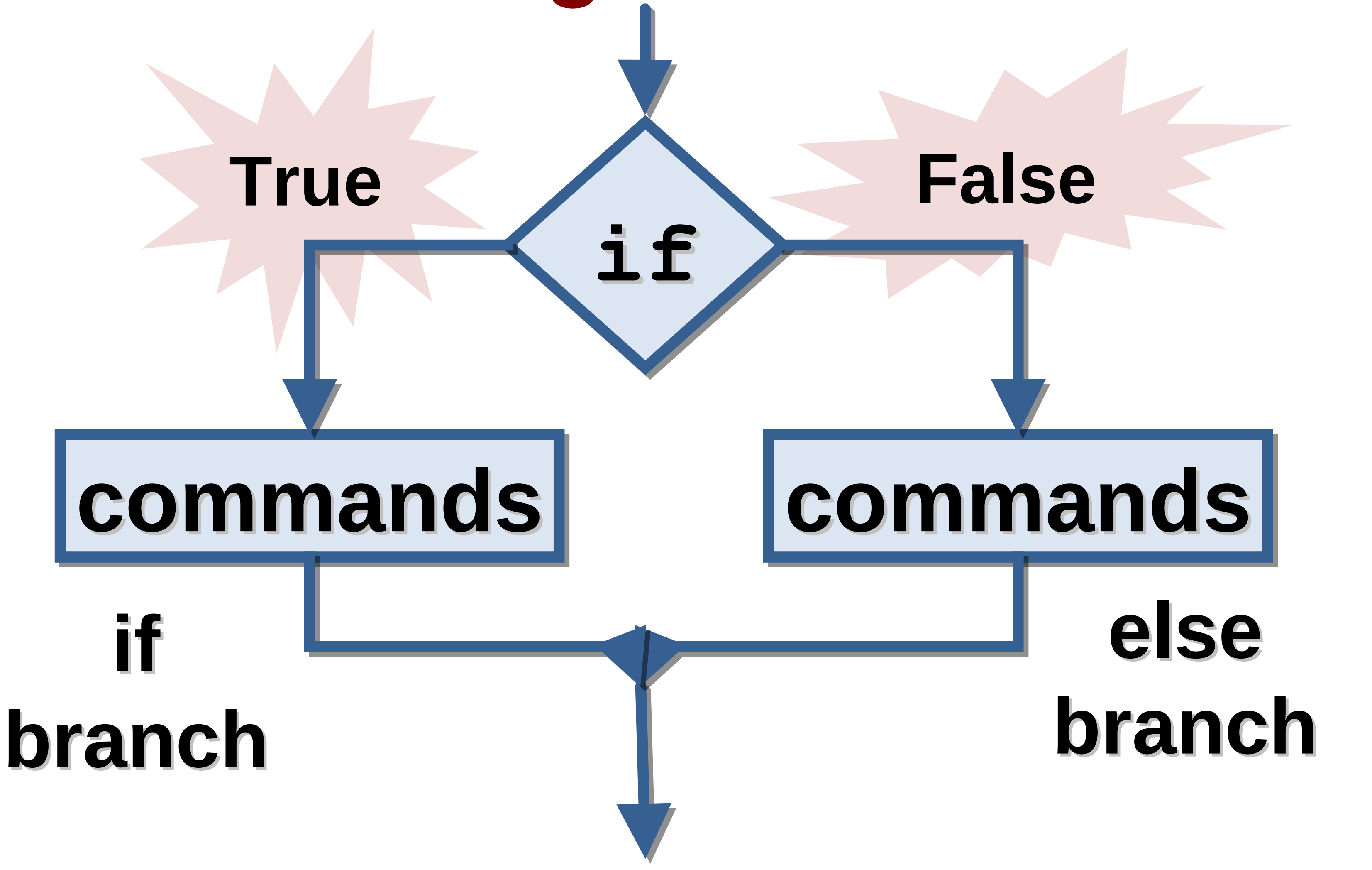
- if ... then ... f i
- if ... then ... else ... f i
- if ... then ... elif ... elif ... else ... fi

```
if [[ <   art> ]]; then
    komutlar
elif [[ <   art> ]]; then
#opsiyonel
    komutlar
else
#opsiyonel
    komutlar
f i
```

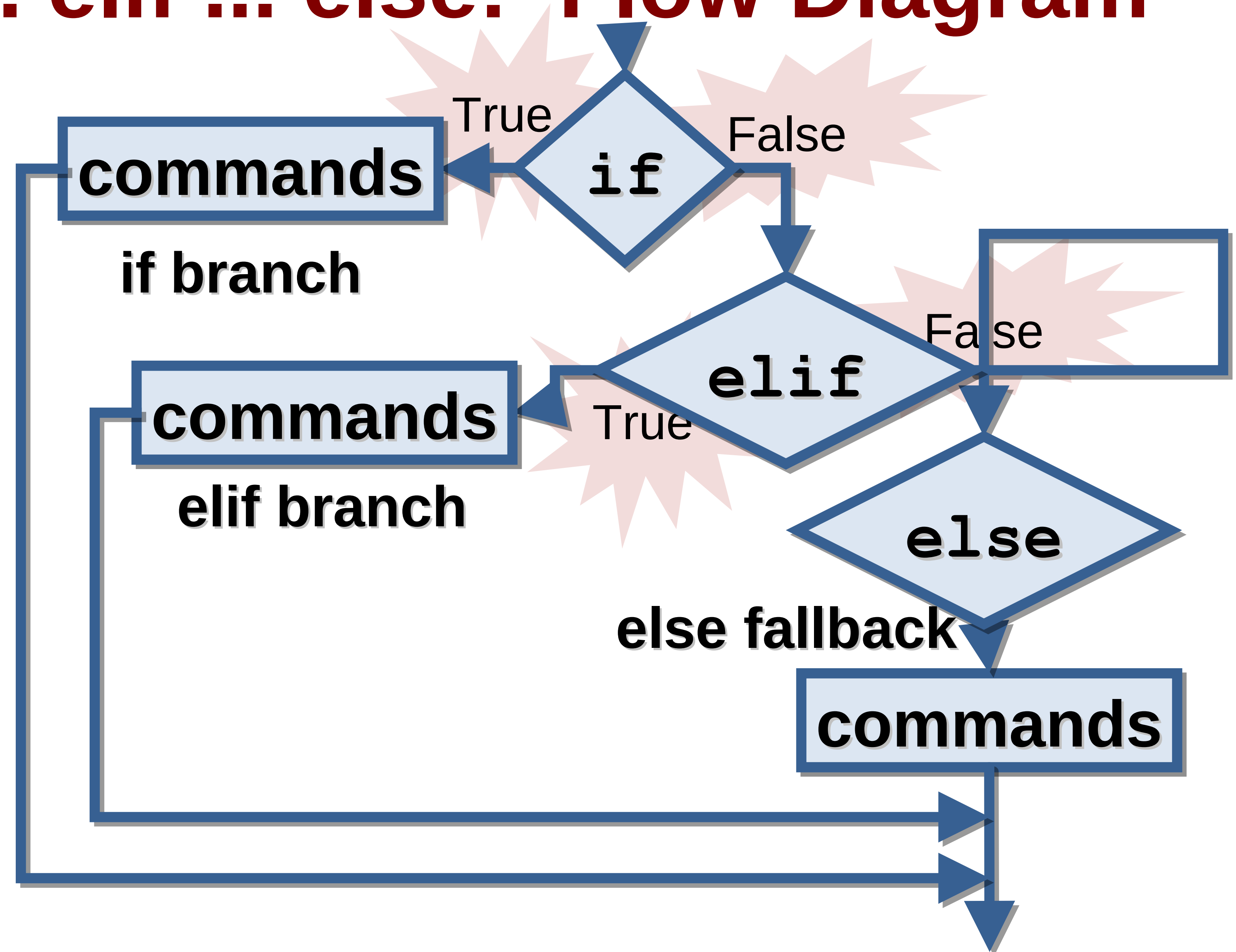
if ... then: Flow Diagram



if … then … else: Flow Diagram



if ... elif ... else: Flow Diagram



String Kar   la   t rma

- ==, = :e   it
- !=   :e   it de   il
- <   :küçük
- >   :büyük
- -n s1 :string s1 bo de   il
- -z s1 :string s1 bo

- #!/bin/bash
- #Declare string S1
- S1="Bash"
- #Declare string S2
- S2="Scripting"
- if [$S1 = $S2]; then
-       echo "e   it"
- else
-       echo "e   it de   il"
- f i

String Kar   la   t rma-2

- if [[$name == “Ali”]]; then
- echo Ali, Oda numaras 12 atand n
- elif [[$name == “Hasan”]]; then
-       echo Hasan, Oda numaras   3 atand n
- elif [[-z $name]]; then
-       echo Ad n z   soylemediniz
- else
-       echo Ad n z$name
- f i

Örnek

```
if [ $fruit = "elma" ] then echo " Elmalar..."
elif [ $fruit = "armut"] then echo " Armutlar..."
elif [ $fruit = "muz" ] then echo " Muzlar"
else echo " Geriye Portakal kald  !"
f i
```

Aritmetik Kar   la t rma

-lt	<
-gt	>
-le	<=
-ge	>=
-eq	==
-ne	!=

```
#!/bin/bash
# declare integers
NUM1=2
NUM2=2
if [ $NUM1 -eq $NUM2 ]; then
        echo "De   erler birbirine e  it"
else
        echo "De   erler birbirine e  it
de  il"
fi
```

Dosya Kar   la   t rmalar

- -e <f le>   : dosya bulunur(exist)
- -f <f le>   : s radan –regular- dosyad r
- -s <f le>   : bo   olmayan (boyu >0) dosyad r
- -d <dir>   : klasördür.
- -w   : dosyaya yazma hakk   vard r.
- -r   : dosyay   okuma hakk   vard r.
- -x   : dosyay   çal   t rma hakk   vard r.

If Cumleleri

```
#!/bin/bash
if [ -h "$1" ] ; then
   echo "Link: $1"
elif [ -f "$1" ] ; then
   echo "File: $1"
elif [ -d "$1" ] ; then
   echo "Directory: $1"
else
   echo "Not link, f le or
dir"
f i
```

-
  ```
  #!/bin/bash
  directory="./BashScriptin
  g"

  # bash klasörmü diye
  kontrol eder
  if [ -d $directory ]; then
  echo «Klasör var"
  else
  echo «Klasör yok"
  f i
  ```

Örnek

```
#!/bin/bash
echo -n "Enter f le name> "
read f le
# Use elif in bash for the “else if” construct.
# The “>>” in the example is output redirection with appending.
# The output of the ls command will be appended to the f le.
if [ -w "$f le" ]; then
    ls >> $f le
    echo "More input has been appended"
elif [ -e "$f le" ]; then
    echo "You have no write permission on $f le"
else
    echo "$f le does not exist"
f i
```

Örnek

```
#!/bin/bash
echo -n "Enter f le name> "
read f le
if [ ! -e $f le ]; then
    echo "Sorry, $f le does not exist."
elif [ ! -w $f le ]; then
    echo "You have no write permission on $f le“
    if [ -o $f le ]; then
        chmod u+w $f le #(grant write permission)
        echo "Write permission granted"
    else
        echo "Write permission cannot be granted"
        echo "because you don't own this f le"
    f i
else
       ls >> $f le
    echo "More input has been appended"
f i
```

Örnek

```
if [-d mydir ]; then
if [ -f the_f le ]; then
        cp the_f le mydir
f i
else
       mkdir mydir
       echo “Klasör mydir yarat  ld”
f i
```

If-else Örnek

```
#!/bin/bash

# Degi   ken tan  mla ve4 de   eri ata
choice=4

echo "1. Bash"
echo "2. Scripting"
echo "3. Tutorial"
echo -n "Seçiminiz [1,2 or 3]? "
# Loop while the variable choice is equal
4
# bash while loop
while [ $choice -eq 4 ]; do

# read user input
read choice
# bash nested if/else
if [ $choice -eq 1 ] ; then

echo "You have chosen word: Bash"
```

```
else

if [ $choice -eq 2 ] ; then
echo "You have chosen word: Scripting"
else

if [ $choice -eq 3 ] ; then
echo "You have chosen word: Tutorial"
else
echo "Please make a choice between 1-3
!"
echo "1. Bash"
echo "2. Scripting"
echo "3. Tutorial"
echo -n "Please choose a word [1,2 or
3]? "
choice=4
f i
f i
f i
done
```

Örnek-   lemler

```
#!/bin/bash
if [[ $1 > 0 && $(($2 % 10)) != 0 ]]; then
    echo Operands are valid
    let “a = $2 % 10”
    let “r = $(($1 * $2)) / $a”
    echo "expression value is $r"
else
    echo "Operand problem"
f i
```

Örnek

```
• difference=$(( $a - $b ))
• echo "The difference a - b is $difference"
• product=$(($a * $b))
• echo "The product a * b is $product"
• if [[ $b -ne 0 ]]; then
•   quotient=$(($a / $b))
•   echo "The division a / b is $quotient"
• else
•   echo "The division a/b is impossible"
• f i
```

Ornek-2 Geli   tirilmi

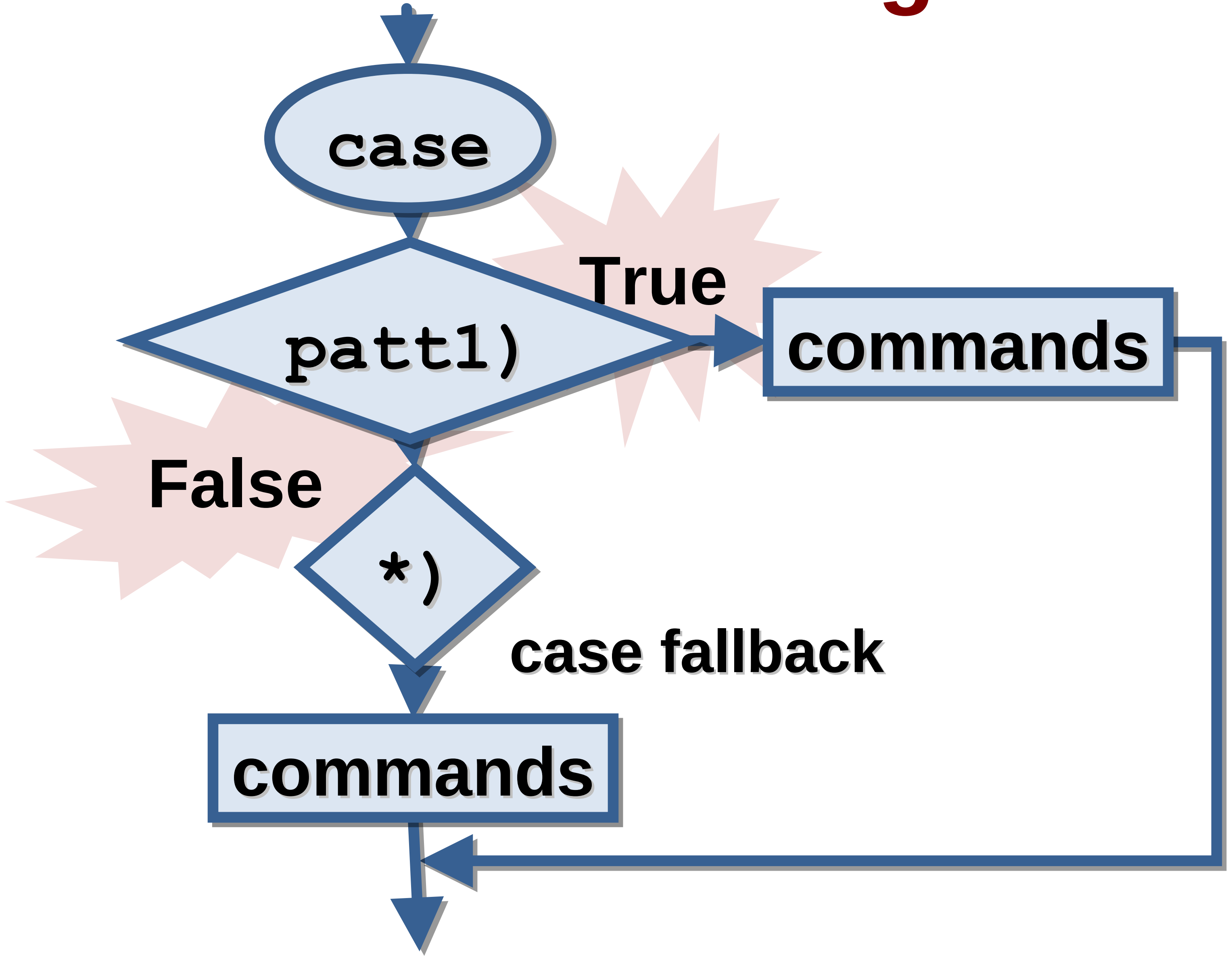
```
#!/bin/bash
if [ $# != 2 ]; # or, if ( test $# != 2 )
then
  echo “Usage: $0 integer1 integer2”
else
      echo “Doing arithmetic>  “
      r=$(($1 + $2)) ; echo “the sum  "$1" + "$2“  is $r”
      r=$(($1 - $2)) ; echo "the subtraction "$1" - "$2" is $r“
      r=$(($1 * $2)) ; echo "the product $1 * $2 is $r“
      if [ $2 -ne 0 ] ; then
                r=$(($1 / $2)) ; echo "the division $1 / $2 is $r"
      else
                echo "the division $1 / $2 is impossible"
      f i
f i
```

Case

```
case expression in
    pattern1 )
       statements ;;
    pattern2 )
       statements ;;
    pattern3 )
      statements ;;
esac
```

```
echo -n "Where do you want
to go? "
read room
case "$room"   in
   "cave")
               echo "It is dark!" ;;
   "hill")
                echo "Tough
climb!" ;;
   "cliff")
                echo "I’m falling!" ;;
   *)
               echo "Can’t go
there!" ;;
```

case: Flow Diagram



Case - Örnek

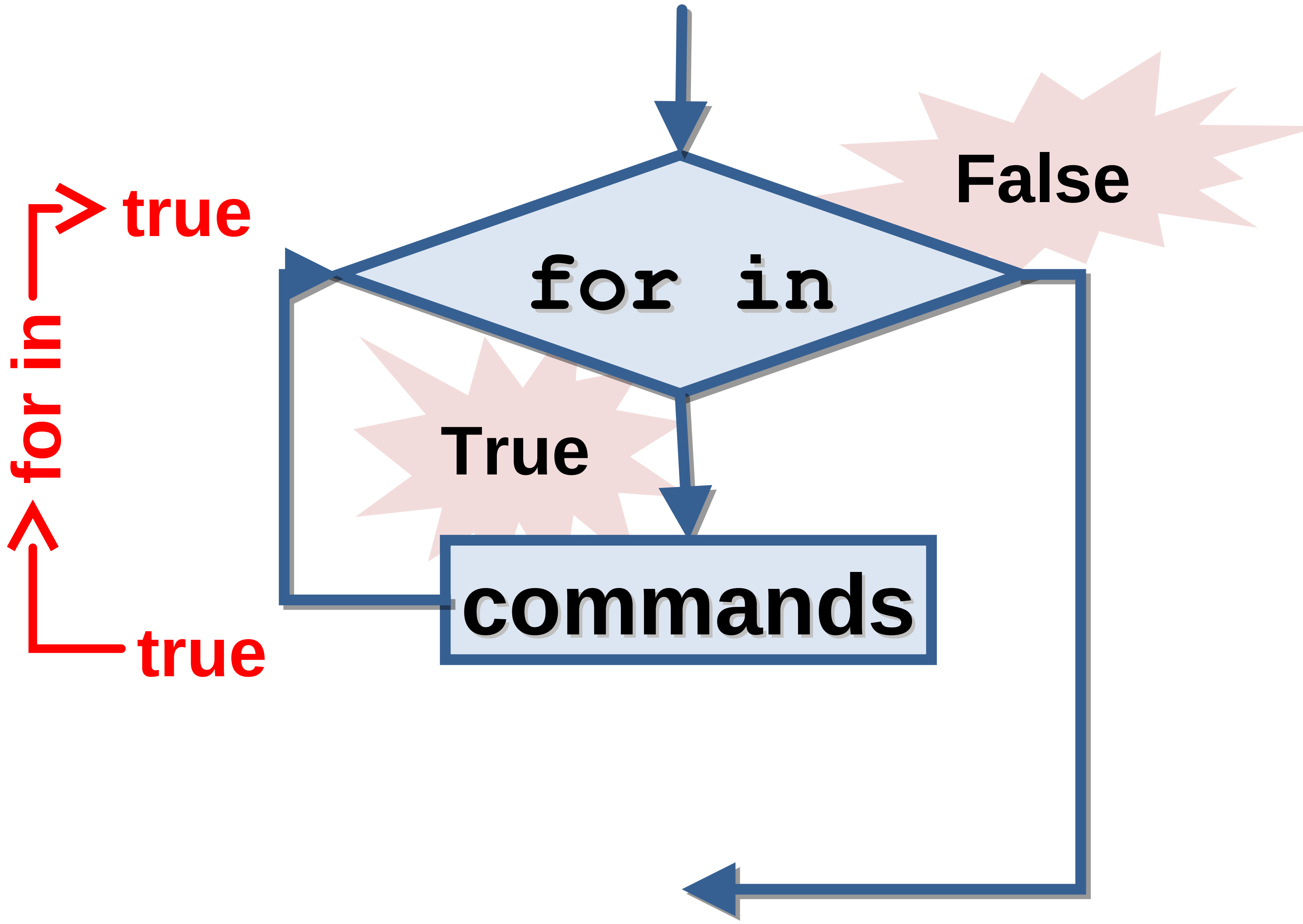
```
case $f lename in
    *.f)
    echo “Fortran 77 source kaynak dosyas ”
    ;;
    *.c)
    echo “C source kaynak dosyas ”
    ;;
    *.py)
    echo “Python script dosyas ”
    ;;
    *)                  #optional, indicates default
    echo  “Bu dosya türünü bilmiyorum”
    ;;
esac
```

For Loop

- lk kullan m   ekli:

```
for degisken in iterator
   do
       komutlar
   done
```

for ... in: Flow Diagram



For Döngüsü Kullanma   ekilleri

-
  ```
  for VAR in {VAR value list} ; do
          { code }
  done
  ```

-
  ```
  for (( i=0; i<5; i++ )) do
          { code }
  done
  ```

-
  ```
  for people in $1 $2 $3 $4; do        # using command line arguments
          echo $people
   done
  ```

-
  ```
  for people in $*; do # using all command line arguments
          echo $people
   done
  ```

Örnekler

- for i in 1 2 3 4 5 ; do
  ```
        echo “I am on step $i”
   done
  ```
- for i in {1..5}
  ```
   do
        echo “I am on step $i”
   done
  ```
- for i in 0{1..9} {10..100}; do
  ```
      echo “File extension will be $i”
   done
  ```

For Loop

- kinci Kullan m   ekli:

```
for (( EXPR1; EXPR2; EXPR3 ))
do
    komutlar
done
```

- Örnek:

```
for (( i=0; i<$IMAX;i++ )); do
    echo $name”.”$i
done
```

For loop

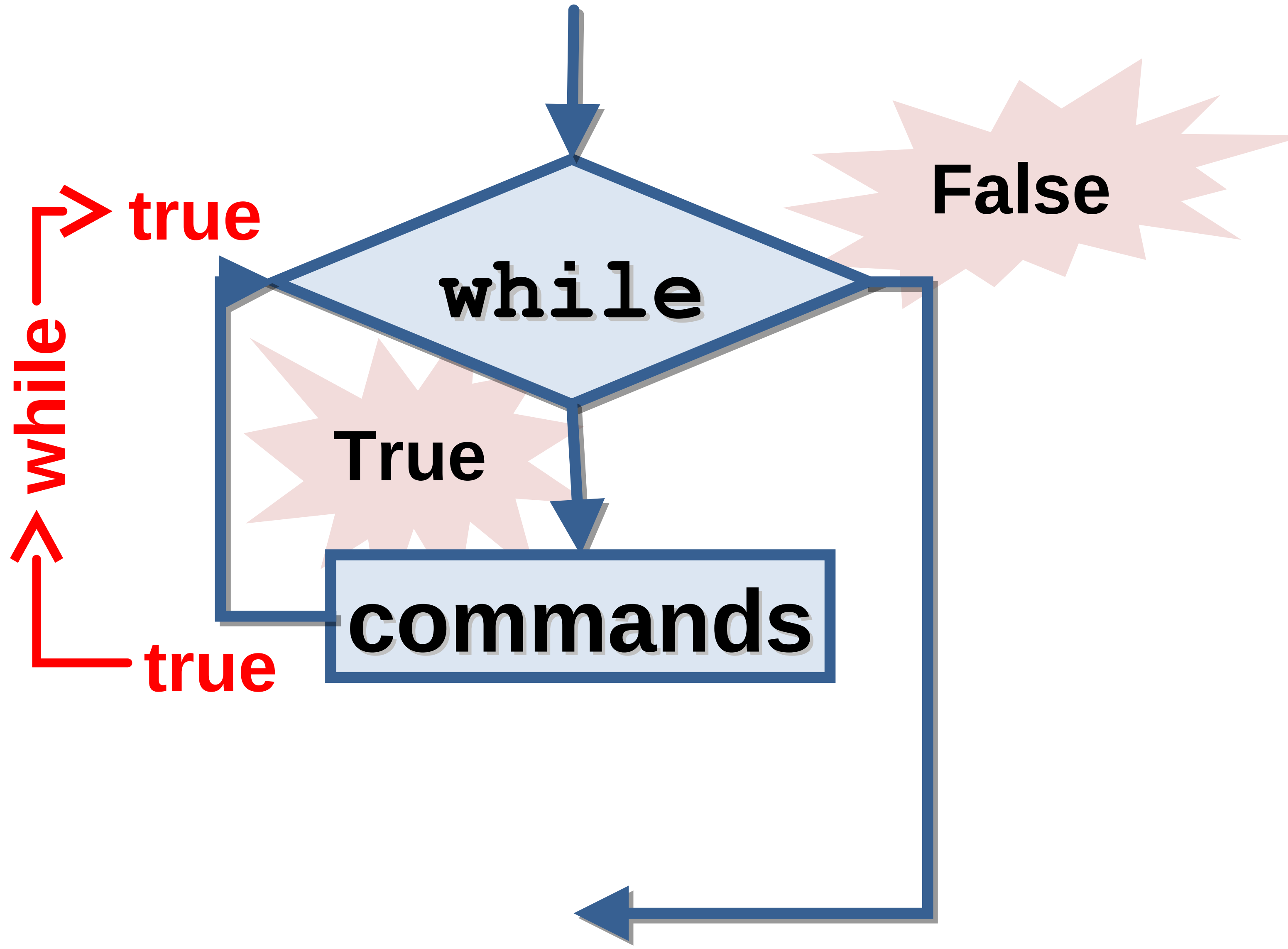
-
  ```
  #!/bin/bash

  # bash for loop
  for f in $( ls /var/ ); do
  echo $f
  done
  ```

While

```
while [ condition ]
do
    komut
    komut
    komut
done
```

-
  ```
  #!/bin/bash
  COUNT=6
  # bash while loop
  while [ $COUNT -gt
  0 ]; do
    echo ‘Degeri:’
  $COUNT
       let
  COUNT=COUNT-1
  done
  ```

while: Flow Diagram



Örnek

```
#!/bin/bash
    secretCode=zoom99
    echo -n "Guess the code> "
    read yourGuess
    while [ $secretCode != $yourGuess ]; do
    echo "Good guess but wrong, try again"
    echo -n "Enter your guess> "
    read yourGuess
    done
    echo "BINGO!"
    exit 0
```

break

```
while [ condition ]
do
    if [ disaster ]; then
        break
    f i
    command
    command
done
```

```
for idx in {0..9} ; do
   if [ $idx = 4 ] ; then
      break
   f i
   echo $idx
done
```

continue

```
for i in iterator; do
    if [[ something ]]; then
        continue
    f i
    command
    command
done
```

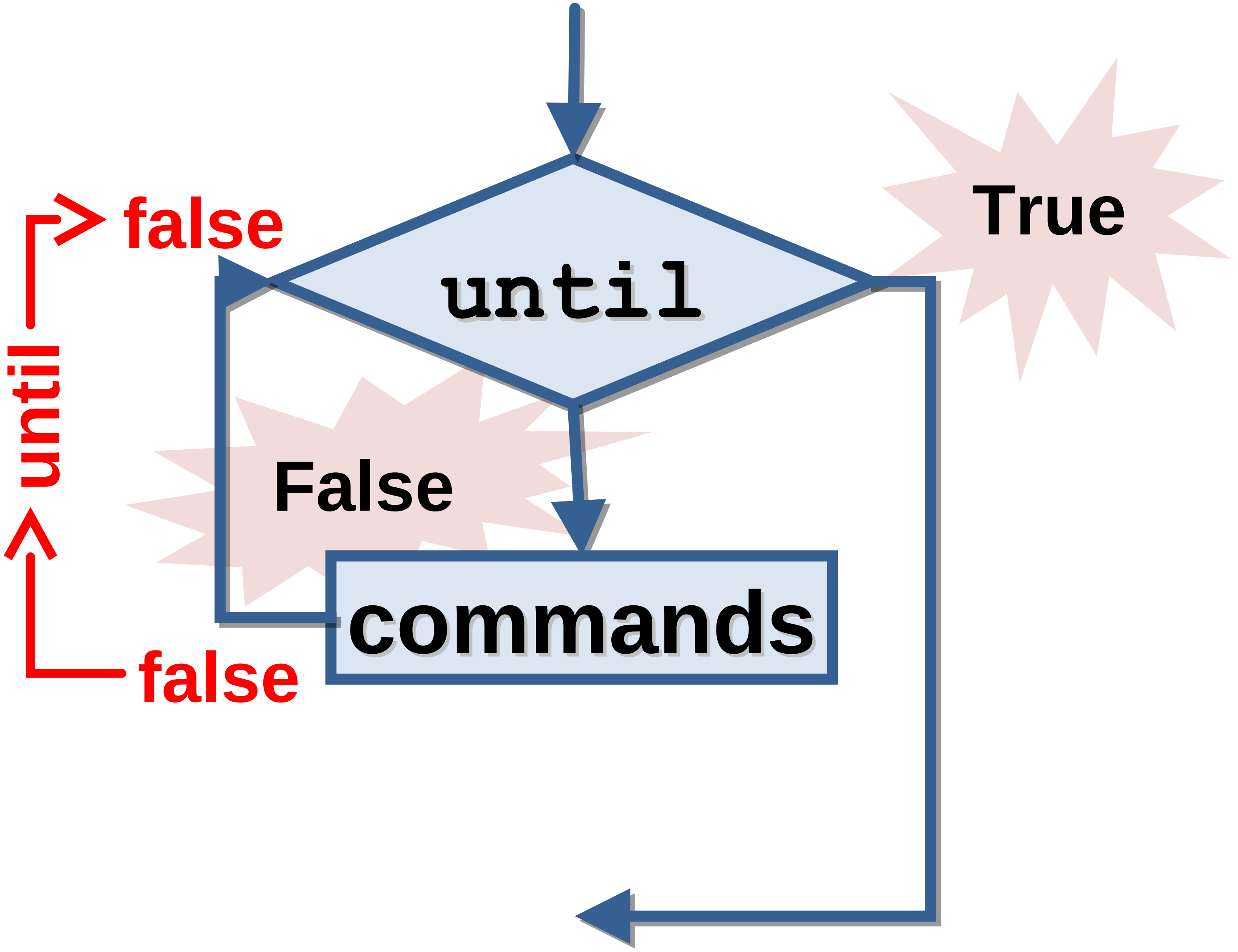
```
for idx in {0..9} ; do
    if [ $idx = 4 ] ; then
        continue
    f i
    echo $idx
done
```

Until

-
  ```
  until [ i -eq 10 ]
  do
     {code}
     let i++
  done
  ```

-
  ```
  #!/bin/bash
  COUNT=0
  # bash until loop
  until [ $COUNT -gt
  5 ]; do
  echo Value of count
  is: $COUNT
  let
  COUNT=COUNT+1
  done
  ```

until: Flow Diagram



```
• secret=4
• guess=""
• echo "Number between 1 & 10"
• until [ "$guess" = "$secret" ] ; do
•     echo -n "Your guess: "
•     read guess
• done
• echo "You guessed it!"
```

Foreach

- foreach f le($argv)
-       sort $f le > $f le.tmp
-       mv $f le.tmp $f le
- end

Özet

- if … then … f i
- if … then … else … f i
- if … then …elif … else … f i
- for … in … do … done
- while … do … done
- until … do … done
- case … in … esac

Bash Aritmetik

- x=$((4+20))
- i=$(($i+1))
- x=1
- x=$[$x+1]
- y=$((2*$x+16))
- Veya ileri aritmetik için bc de kullan labilir:
- x=$(echo “3*8+$z” | bc)
- Veya ‘expr’ da kullan labilir:
- x=`expr $x + 1`   # x i 1 art r r.
- x=$(expr $x * 2)

Aritmetik ve Mant ksal

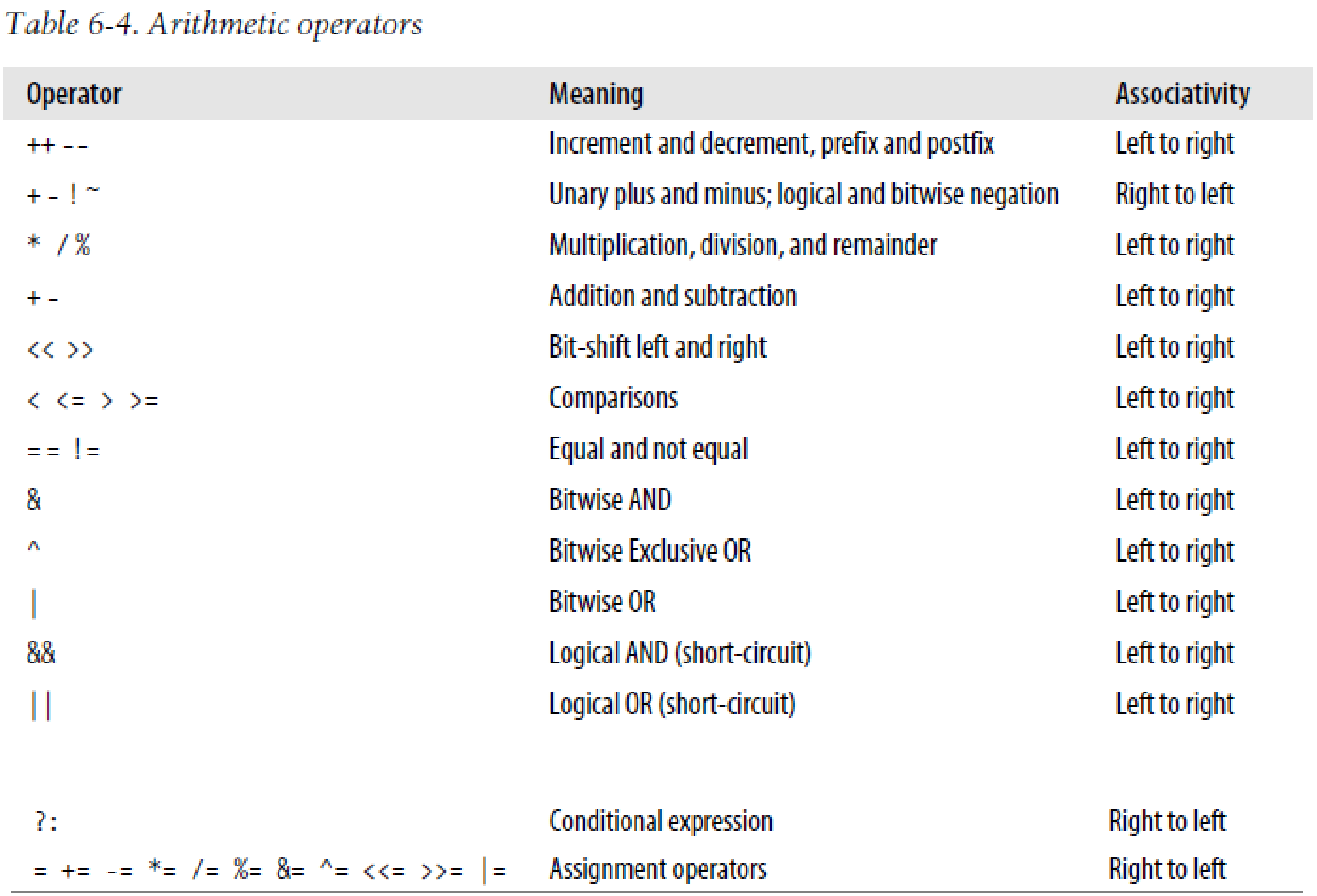


Table 6-4. Arithmetic operators

Operator	Meaning	Associativity
`++ --`	Increment and decrement, prefix and postfix	Left to right
`+ - ! ~`	Unary plus and minus; logical and bitwise negation	Right to left
`* / %`	Multiplication, division, and remainder	Left to right
`+ -`	Addition and subtraction	Left to right
`<< >>`	Bit-shift left and right	Left to right
`< <= > >=`	Comparisons	Left to right
`== !=`	Equal and not equal	Left to right
`&`	Bitwise AND	Left to right
`^`	Bitwise Exclusive OR	Left to right
`|`	Bitwise OR	Left to right
`&&`	Logical AND (short-circuit)	Left to right
`||`	Logical OR (short-circuit)	Left to right
`?:`	Conditional expression	Right to left
`= += -= *= /= %= &= ^= <<= >>= |=`	Assignment operators	Right to left

Komut sat r Argümanlar

- Komut sat r dan girilen argumanlar `$0`, `$1`, `$2`, vb. le belirtilir:
- $0 script-komut ad d r
- Di er argumanlar s rayla girilen parametrelerdir.
- $# arguman say s n verir.
- ./hello.sh ali hasan ayse

```
#!/bin/bash
USAGE="Usage:$0 dir1"
if [ "$#" == "0" ]; then
    echo "$USAGE"
    exit 1
f i
while [ $# -gt 0 ]; do
    echo “$1”
done
```

Fonksiyonlar

```
function name() {
      komutlar
      komutlar
      komutar
           VALUE=sayi
      return $VALUE
      }
```

```
#!/bin/bash
function writeout() {
   echo $1
}
writeout “Hello World”
```

Örnek

```
#!/bin/bash
#Global de   i   ken
myvar="hello"
myfunc() {
    myvar="one two three"
    for x in $myvar
    do
        echo $x
    done
}
myfunc  echo $myvar $x
```

```
#!bin/bash
#Lokal de   i   ken
myvar="hello"
myfunc() {
   local x
   local myvar="one two three"
   for x in $myvar ; do
      echo $x
   done
}
myfunc echo $myvar $x
```

Örnek

```
#!/bin/bash
if [ $# -ne 2 ]       # Argument check
        then echo "Usage: $0 f irst-number second-number"
        exit 1
f i
gcd () {
  dividend=$1; divisor=$2; remainder=1
  until [ "$remainder" -eq 0 ]
  do
    let "remainder = $dividend % $divisor"
    dividend=$divisor; divisor=$remainder
  done
}
gcd $1 $2
echo; echo "GCD of $1 and $2 = $dividend"; echo
```

String   lemleri

- Concatenation - Birleştirme

  `newstring=$oldstring”.ext”`

- String boyu

  `${#string}`

- Substring çıkarma
  - İlk karakter 0 olarak numaralanır
  - `pos` sırasından sonuna kadar

    `${string:pos}`
  - `pos` sırasından `len` boyuna kadar

    `${string:pos:len}`

String   lemleri-2

- Delete shortest match from front of string

```
${string#substring}
```

- Delete longest match from front of string

```
${string##substring}
```

- Delete shortest match from back of string

```
${string%substring}
```

- Delete longest match from back of string

```
${string%%substring}
```

Array -Diziler

- Diziye eleman () i   aretleri ile girilir, ${ } i   aretleri ile diziden okunur. Örnek:

- names=(zmir Ankara   stanbul)
- echo ${names[0]}   #   zmir yazar
- echo ${names[@]:1:2}   # Ankara   stanbul yazar
- echo ${names[*]}   #   zmir Ankara   stanbul yazar
- echo ${#names[@]}   #eleman say s n   yani 3 yazar
- names=(${names[@]} Antalya)   #yeni bir eleman ekler
- names[1]=Quebec   # 1. dizi eleman n   (2. f iziksel eleman)de   i   tirir
- echo $names[@]   #   zmir Quebec   stanbul Antalya yazar

Örnekler

```
my_arr=(1 2 3 4 5 6)
for num in $
{my_arr[@]}; do
    echo $num
done
```

```
jpg_files=(`ls *jpg`)
for file in $
{jpg_files[*]}; do
   if [[ -n $file ]];
then
    convert $file $
{file%%”.jpg”}.png
   fi
done
```

```
names=(   zmir Ankara   stanbul)
for name in $*; do
       array=("${array[@]}" $name)
done

    echo ${array[@]} #print all the array elements
    i=0
until [ $i -eq $# ]; do
    echo -n ${array[$i]} #print one array element
    echo
    let i++
done
```

- #!/bin/bash
- echo -e "Merhaba, kelime girin: \c "
- read word
- echo "Girdiginiz kelime: $word"
- echo -e "Iki kelime girermisiniz? "
- read word1 word2
- echo "Girdikleriniz: \"$word1\" \"$word2\""
- echo -e " bash scripting hakkinda dusundukleriniz? "
- #   eger read bos girilirse, $REPLY adli degiskenden saklanir
- read
- echo "Siz $REPLY dediniz, Tesekkurler! "
- echo -e "Favori renginiz ? "
- # -a parametresi read komutunda diziye okutur
- read -a colours
- echo "Renkleriniz ${colours[0]}, ${colours[1]} ve ${colours[2]}:-)"

- #!/bin/bash
- #Declare array with 4 elements
- ARRAY=('BLP101' 'BLP102' ‘BLP103’)
- # dizi eleman sayisi…
- ELEMENTS=${#ARRAY[@]}

- # echo her dizi elemani
- # for loop
- for ((i=0;i<$ELEMENTS;i++)); do
-     echo ${ARRAY[${i}]}
- done

Shell Script Çevre De   i   kenleri

```
export EDITOR=emacs
if [[ $REMOTEHOST ]]; then
 if [[ $REMOTEHOST == "csgate.uwindsor.ca“ ]]; then
  export DISPLAY="U96.lamf.uwindsor.ca:0.0"
 else
  export DISPLAY=$REMOTEHOST":0.0"
 f i
else
 export DISPLAY=$HOST":0.0"
f i
```