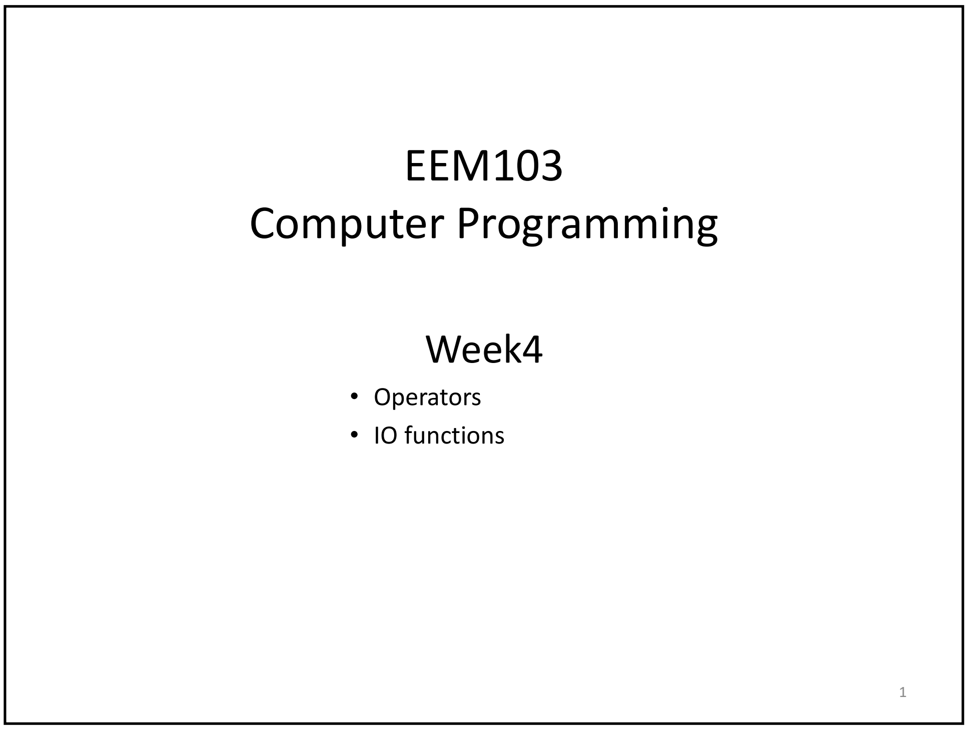
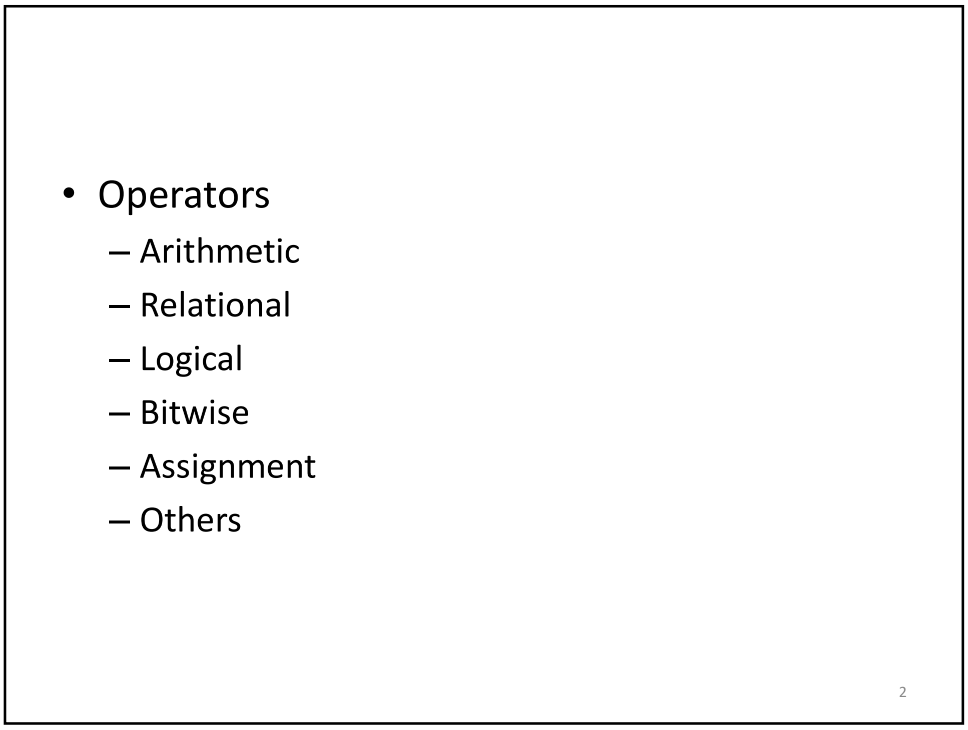


EEM103 Computer Programming

Week4

- Operators
- IO functions

1



- Operators
  - Arithmetic
  - Relational
  - Logical
  - Bitwise
  - Assignment
  - Others

2

Arithmetic operators

- Double operand:

  < operand> < operator> < operand>

  + addition

  - subtraction

  ***** multiply

  / division

  % mod (remainder – bölümünden kalan)

  8%3→2, 8%4→0, 8%8→0, 8%10→8

Division

/ division

10 / 4 → 2

10.0 / 4 → 2.5   10 / 4.0 → 2.5   10.0 / 4.0 → 2.5

```
int number ;
number = 10.0/4.0;   → number =2

float a;
int b=10, c=4;
a = b / c;   →   a=2;

a = b / c.0 ;  → ☠
```

Division

float a;
int b=10, c=4;

a = b*1.0 / c; → a=2.5

a = b / (c*1.0); → a=2.5

a = **(float)** b / c ; → a=2.5 CASTİNG

a = b / **(float)** c ; → a=2.5 CASTİNG

5

- Single operand:
  <operator> < operand> (or vice versa)
  ++ increment by 1,
  -- decrement by 1,

- Pre-fix vs. Suf-fix;
  The place of increment/decrement operator changes the order of process <u>if there are more than one operation in a statement</u>.

```
int a=5 , b;
b = a++;                          → assign a to b, then increment a.
printf("a=%d , b=%d", a,b );      → a=6 , b=5

int a=5 , b;
b = ++a;                          → increment a, then assign a to b.
printf("a=%d , b=%d", a,b );      → a=6 , b=6
```

6

Relational operators

>	greater than
>=	greater than or equal
<	less than
<=	less than or equal
==	equal to
!=	not equal to

- The result of a relational operator is either **1 (True)** or **0 (False)**.

7

Logical Operators

&&	VE (AND)
||	VEYA (OR)
!	DEĞİL (NOT)

The result of a logical operator is either 1 (True) or 0 (False).

a	b	a && b	a || b	!a
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

0 is logically **FALSE (0)**,
All other numbers which are not equal to 0 is **TRUE(1)**

18 && 42 → 1 && 1 → 1
18 && 0 → 1 && 0 → 0
!6 → !1 → 0

8

expression1	expression2	expression1 && expression2
0	0	0
0	nonzero	0
nonzero	0	0
nonzero	nonzero	1

expression1	expression2	expression1 || expression2
0	0	0
0	nonzero	1
nonzero	0	1
nonzero	nonzero	1

expression	!expression
0	1
nonzero	0

Bitwise operators

- The bitwise operators are used to manipulate the bits of operands both `signed` and `unsigned`.
  - Unsigned integers are normally used with the bitwise operators.
- The bitwise operators are
  - bitwise AND (&),
  - bitwise inclusive OR (|),
  - bitwise exclusive OR (^; also known as bitwise XOR),
  - left shift (<<),
  - right shift (>>) and
  - complement (~).

Bit 1	Bit 2	Bit 1 & Bit 2
0	0	0
0	1	0
1	0	0
1	1	1

Bit 1	Bit 2	Bit 1 | Bit 2
0	0	0
0	1	1
1	0	1
1	1	1

Bit 1	Bit 2	Bit 1 ^ Bit 2
0	0	0
0	1	1
1	0	1
1	1	0

- Problem

  6 & 8 = ?

  6 && 8 = ?

  6 | 8 = ?

11

Bitwise shift operators

>> shift to the right (divide by 2)

<< shift to the left (multiply by 2)

E.g: 6 >> 1 (shift right 1 bit)

0000 0$\overrightarrow{11}$0 >> 1

0000 0011 = 3

E.g: 6 << 2 (shift left 2 bit)

000$\overleftarrow{0\ 01}$10 << 2

0001 1000 = 24

12

Assignment operators

= Assignment operatör

Left-value ← Right-value :

R-value may contain more than one operation.

L-value should contain only one variable.

E.g: a = b+c → True, a=b+c+d*(e+f) → True

a+b = c+d → False

Assignment operators

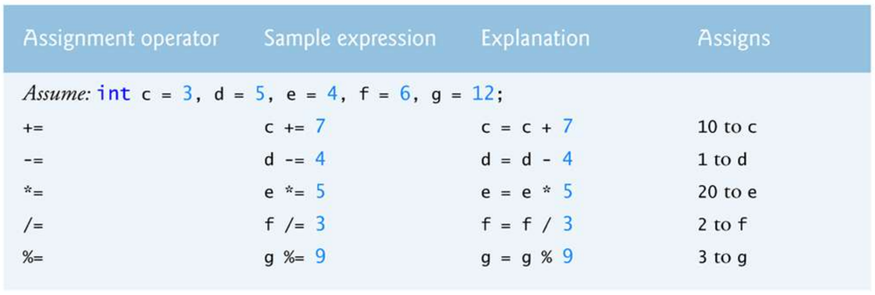
Assignment by operation

+= -= *= /=

%= <<= >>= &= |=

+= → assignment by addition

E.g. a += 5 → a = a+5



Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 to c
`-=`	`d -= 4`	`d = d - 4`	1 to d
`*=`	`e *= 5`	`e = e * 5`	20 to e
`/=`	`f /= 3`	`f = f / 3`	2 to f
`%=`	`g %= 9`	`g = g % 9`	3 to g

15

Other operators

- **sizeof()**
  - returns the size of a variable in memory (in bytes)

    sizeof(int) → 4

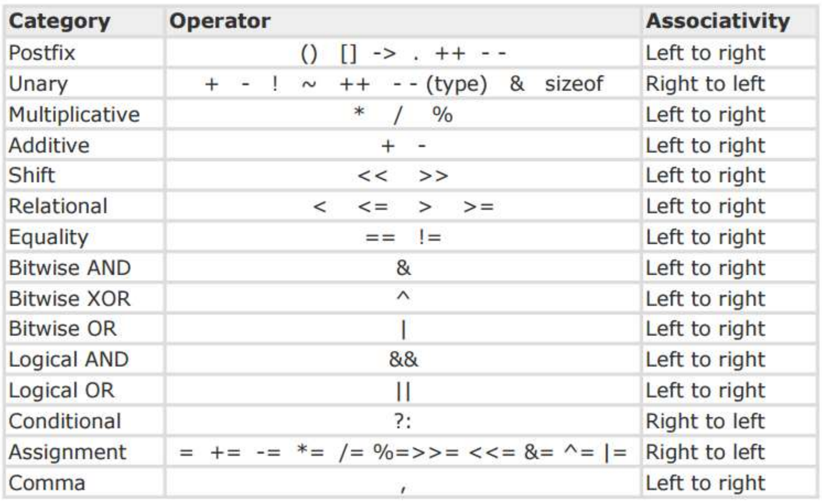
    char c ;

    sizeof(c) → 1

*	content of	(in pointers chapter)
&	address of	(in pointers chapter)
? :	Conditional Expression	(in control statements chapter)

16

Operators Precedence



Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type) & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

17