Lecture - 07

Programming in C

⇒ Operators & Expressions

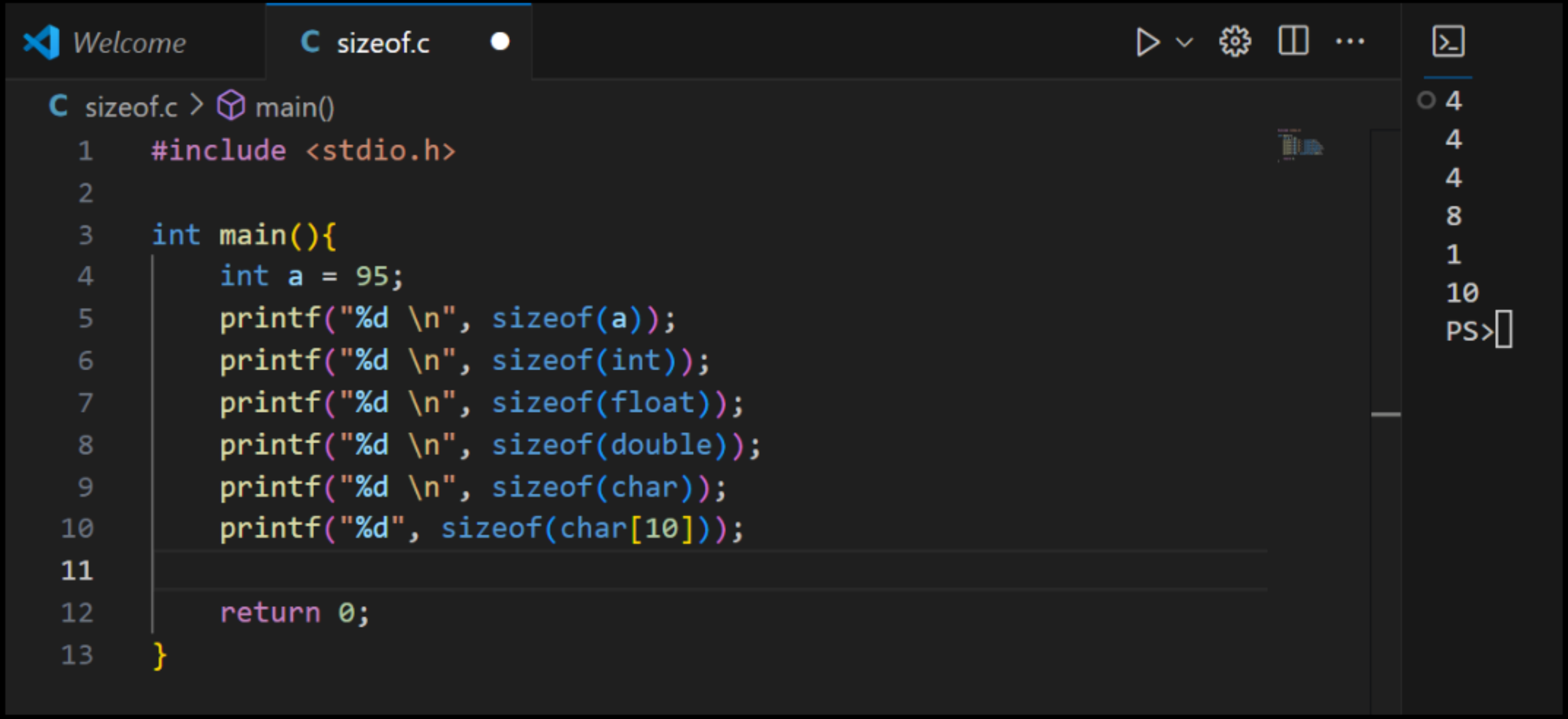
Sizeof ( ) function in C ← Operator

↳ return the size of a data item
↳ in bytes
└ int, float, double... etc

```
int x = 75;

printf("%d", sizeof(x))
```

$\underbrace{\text{sizeof}}$ ↳ size of variable x = 4 ← int



```c
#include <stdio.h>

int main(){
    int a = 95;
    printf("%d \n", sizeof(a));
    printf("%d \n", sizeof(int));
    printf("%d \n", sizeof(float));
    printf("%d \n", sizeof(double));
    printf("%d \n", sizeof(char));
    printf("%d", sizeof(char[10]));

    return 0;
}
```

```
4
4
4
8
1
10
PS>
```

Expression

↳ An expression is a Combination of variables, Constants, operators and functions that are evaluated to produce a value.

Eg

variable   literal / Constant

datatype → `int x = 10;`   operator

`int y = 20;`   punctuator

`int sum = x+y;`   expression → x + y

resultant datatype   resultant variable

operator

x + y

Operands

```
print("%d", sum);
```

Types of an expression:

a) Constant Expression : An expression having Constant Value.

eg: 5, A, 3.14, 2.718

b) Variable Expression: An expression with single Variable

Eg x, y, z

c) Arithmetic Expression, Relational Expression, Logical Expression, Assignment Expression

Arithmetic Expression $\downarrow$ (a+b), (a-b)

Relational Expression $\Downarrow$ a<b, x>y

Logical Expression $\Downarrow$ (a>b) && (c<d)

Assignment Expression $\Downarrow$ x = 10, y=20

d) Compound Expression

↳ (a+b) * (c-d)

Operators :

↳ Operators are the symbols that perform an operation on operands.

Ex $x + y$, x, y are operands

$+$ is an operator

Addition is an operation

Types of operators :

a) Unary operator :

↳ perform an operation on single operand.

Eg ++, --, !, ~, $-$ ←

Eg → $x = 15$

$-x$ ← -15

b) binary Operator : An operator perform an operation on two operands.

1) Arithmetic Operators (+ - * / %)
2) Relational Operators (==, !=, <, >, <=, >=)
3) Assignment operators (=, +=, -=, *=, /=, %=)
4) Logical operators (&&, ||, !)
5) Bitwise operators (&, |, ~, ^, <<, >>)

$\uparrow$ XOR

c) Trinary Operator : An operator perform an operation on three operands.

Eg Condition ? Exp1 : Exp2

⇒ Typecasting operators

Type Conversion / Type Casting :

↳ Modifying / Changing the type of data.

Eg

int → float   char → int

float → int   int → char

Syntax ⇒

(type) data

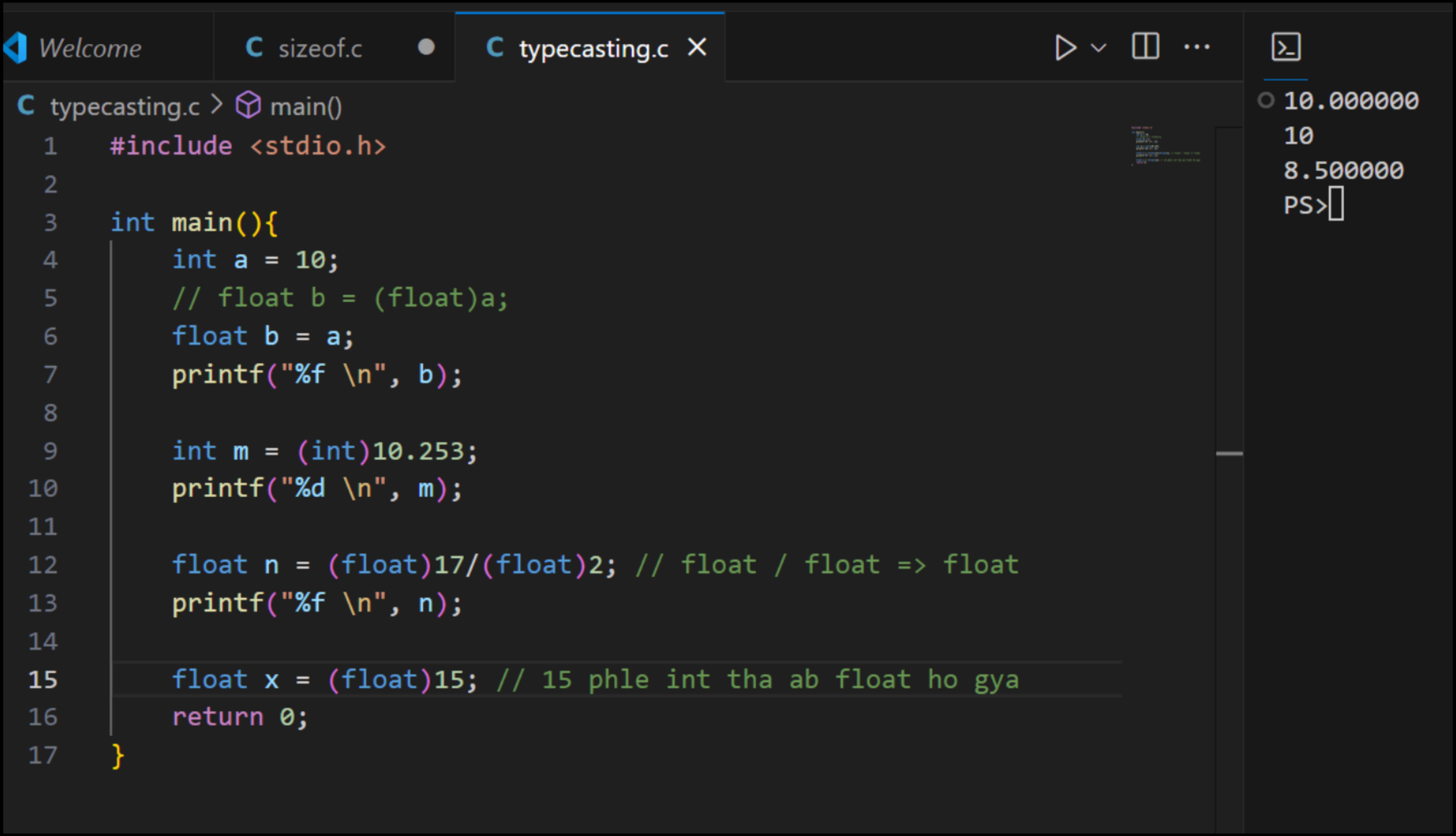
```
int x = 17; ✓
int y = 2; ✓
float d = x / y
printf("%f", d);
```

$\downarrow$ float   float

int/int ⟶ int

$\underbrace{\text{(float) } x/y}_{\text{type conversion}}$;

$\frac{17}{2} = 8.5$ ← float



```c
#include <stdio.h>

int main(){
    int a = 10;
    // float b = (float)a;
    float b = a;
    printf("%f \n", b);

    int m = (int)10.253;
    printf("%d \n", m);

    float n = (float)17/(float)2; // float / float => float
    printf("%f \n", n);

    float x = (float)15; // 15 phle int tha ab float ho gya
    return 0;
}
```

```
10.000000
10
8.500000
PS>
```

(A) Unary operator :

↳ Single operand

Ex int a = 10;

a++; ⟶ increment a by 1

post increment

printf("%d", a); # 11

* Unary Post increment Operator increase the value by 1 at last

int b = 10;

pre increment ++b; ⟵ increase the value by 1 b = 11

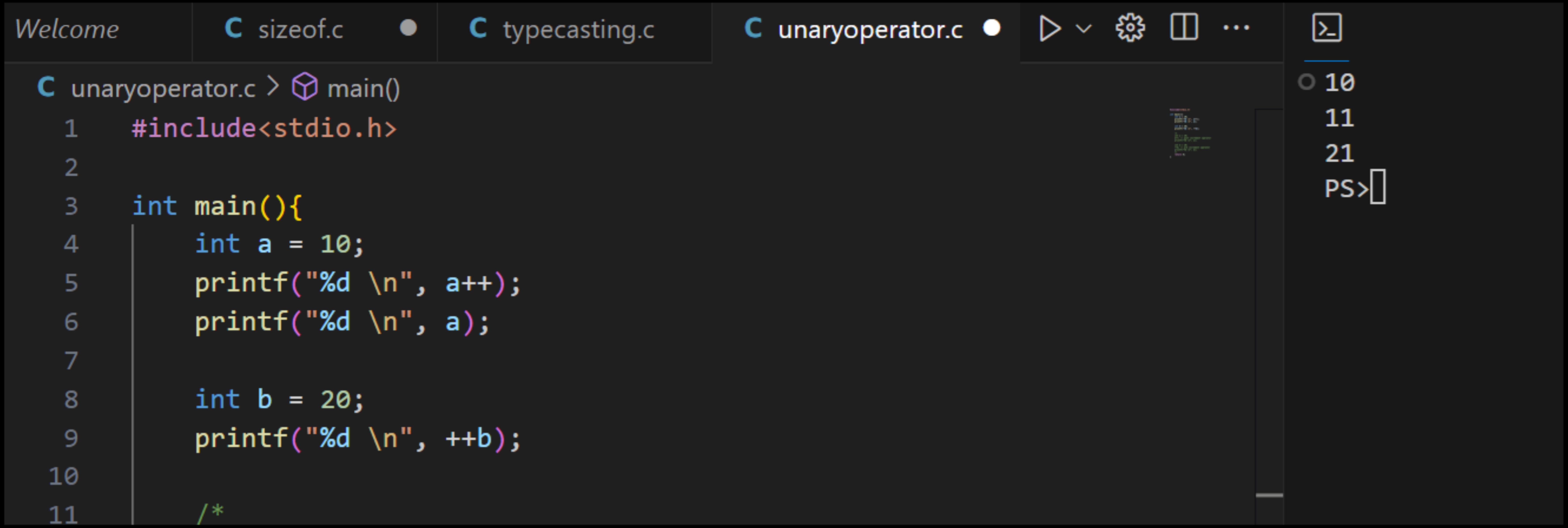
printf("%d \n", a); #

$\underset{\sim\sim}{a++} \leftarrow$ increment करो Last में

$\hookrightarrow$ print the value of 'a' first then increment

$\underset{\sim\sim}{++a} \leftarrow$ increase the value first then print

$\hookrightarrow$ Increment पहले कर दो।



```c
#include<stdio.h>

int main(){
    int a = 10;
    printf("%d \n", a++);
    printf("%d \n", a);

    int b = 20;
    printf("%d \n", ++b);

    /*
```

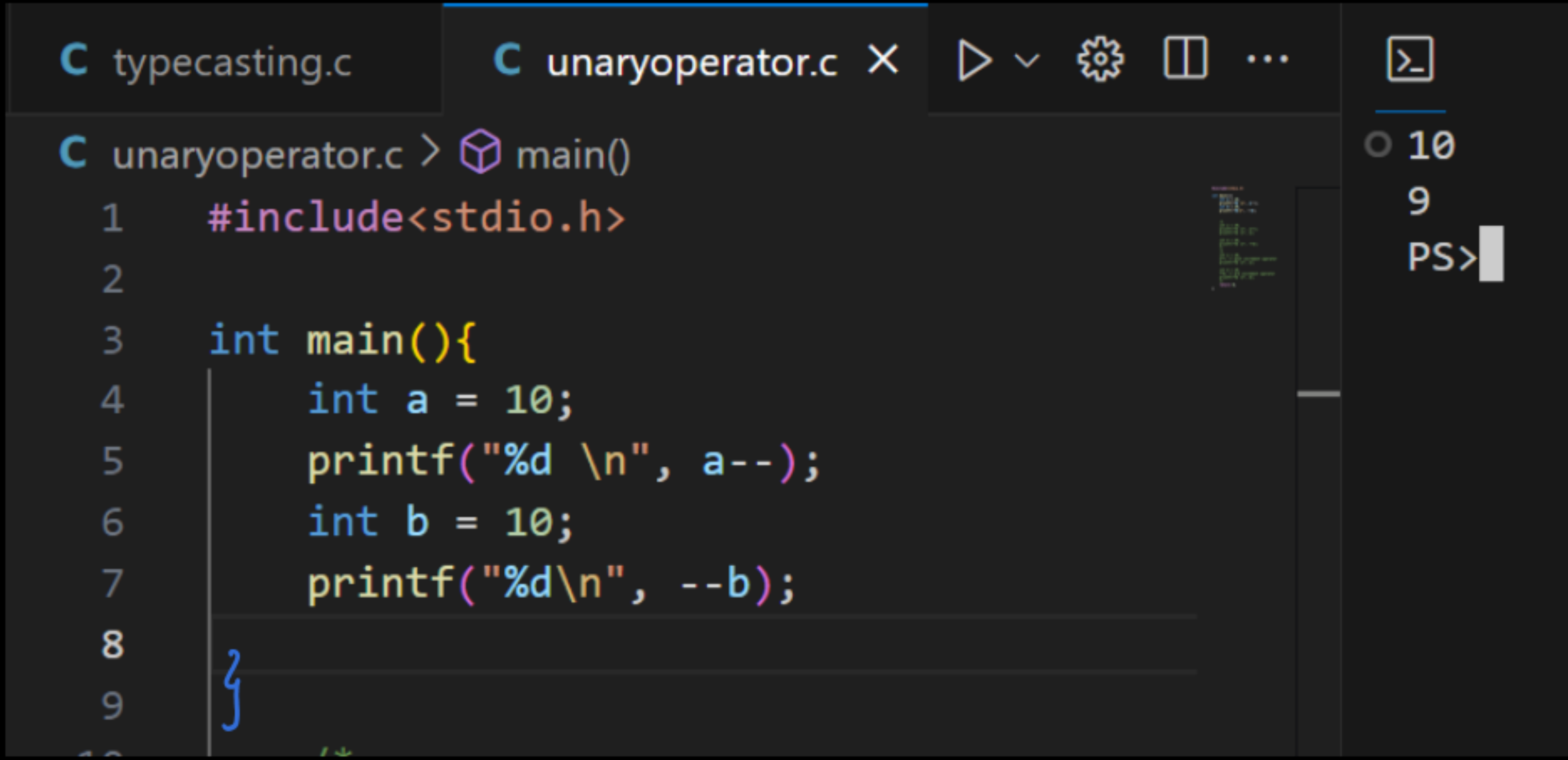
```
10
11
21
PS>
```

Unary decrement operator: Reduce the value by 1.

```
int a = 10;
a-- ;    ← decrease the value by 1, at last
printf("%d", a);
         int   9
```

Post decrement operator

```
int b = 10;
--b;    ← decrease the value by 1, before
printf("%d", b);
        int   9
```



```c
#include<stdio.h>

int main(){
    int a = 10;
    printf("%d \n", a--);
    int b = 10;
    printf("%d\n", --b);
}
```

```
10
9
PS>
```