

PROGRAMMAZIONE PROCEDURALE

A.A. 2025/2026



LITERALS



LITERALS

- ⌚ A *literal* is a token that denotes a fixed value, which may be an **integer**, a **floating-point** number, a **character**, or a **string**.
- ⌚ A literal's type is determined by its value and its notation.

INTEGER CONSTANTS

- ⊙ An *integer constant* can be expressed as an ordinary decimal numeral, or as a numeral in octal or hexadecimal notation.
- ⊙ You must specify the intended notation by a prefix.
 - ✓ **base-8** number system?
 - ✓ 0, 1, 2, 3, 4, 5, 7 $\rightarrow 8^n x + 8^{n-1} + \dots + 8^1 + 8^0$
- ⊙ A *decimal constant* begins with a nonzero digit
- ⊙ A number that begins with a leading zero is interpreted as an *octal constant*. $047 = ?$
 - ✓ $39 = (4 \times 8^1 + 7 \times 8^0)$
- ⊙ A *hexadecimal constant* begins with the prefix 0x or 0X. Hexadecimal digits A to F can be upper or lowercase.
 - ✓ $0xff, 0Xff, 0xFF, \text{ and } 0XFF = 15 \times 16^1 + 15 \times 16^0 = 255$

TYPE OF CONSTANTS

- ⌚ The type of a constant is determined at the same time as its value is defined.
- ⌚ Integer constants such as the examples just mentioned usually have the type **int**. If larger:
 - ✓ The compiler assigns it the first type in a hierarchy that is large enough to represent the value.
- ⌚ For example, a short is 2 byte2, the decimal constant 50000 has the type int, since the greatest possible short value is 32,767, or $2^{15} - 1$.
- ⌚ You can also influence the types of constants in your programs explicitly by using suffixes.
 - ✓ 512U (unsigned int), 0Xf0fUL (unsigned long), 0777ll (long long), 123uLL (unsigned long long)

EXAMPLES

```
int a= 512U;
```

```
int b= 1LL;
```

```
int c= 7UL;
```

Uppercase or lowercase is the same,
e.g., ll or LL

FLOATING-POINT CONSTANTS

- ⌚ Floating-point constants can be written either in decimal or in hexadecimal notation.
- ⌚ A floating-point constant consists of a sequence of decimal digits containing a decimal point.
- ⌚ You may also multiply the value by a power of 10, as in scientific notation: the power of 10 is represented simply by an exponent, introduced by the letter **e** or **E**.
 - ✓ 2.34E5 (2.34×10^5), 67e-12 (67.0×10^{-12})
- ⌚ The decimal point can also be the first or last character. Thus **10.** and **.234E6** are permissible numerals.
 - ✓ However, the numeral 10 with no decimal point would be an integer constant, not a floating-point constant.

FORCING THE TYPE OF FLOATS

- ④ The default type of a floating-point constant is **double**.
- ④ You can also append the suffix **F** or **f** to assign a constant the type **float**, or the suffix **L** or **l** to give a constant the type **long double**

CHARACTER CONSTANTS

Ⓢ A *character constant* consists of one character enclosed in single quotation marks.

✓ 'a' '0' '*'

Ⓢ To represent ', \ and newline

✓ '\" '\\ '\n'

Ⓢ Character constants have the type **int**

ESCAPE SEQUENCES

Escape sequence	Character value	Action on output device
\'	A single quotation mark (')	Prints the character.
\"	A double quotation mark (")	
\?	A question mark (?)	
\\	A backslash character (\)	
\a	Alert	Generates an audible or visible signal.
\b	Backspace	Moves the active position back one character.
\f	Form feed	Moves the active position to the beginning of the next page.
\n	Line feed	Moves the active position to the beginning of the next line.
\r	Carriage return	Moves the active position to the beginning of the current line.

STRING LITERALS

- ⊙ A *string literal* consists of a sequence of characters (and/or escape sequences) enclosed in double quotation marks.
 - ✓ "Hello world!\n"
- ⊙ Like character constants, string literals may contain all the characters in the source character set.
- ⊙ A string literal is a static array of char that contains character codes followed by a string terminator, the null character \0.
- ⊙ The empty string "" occupies exactly one byte in memory, which holds the terminating null character.

```
char helloWorld[128] = "Hello World!\n";  
printf("Print the string: %s\n", helloWorld);
```

SU LIBRO

- ④ Sezione 15.6
- ④ Pagina 339
- ④ Pagina 46, sezione 9.10

