Linguaggi di programmazione

- **Il "potere espressivo" di un linguaggio** è caratterizzato da:
- **quali tipi di dati** consente di rappresentare (direttamente o tramite definizione dell'utente)
- **quali istruzioni di controllo** mette a disposizione (quali operazioni e in quale ordine di esecuzione)

PROGRAMMA = DATI + CONTROLLO

Linguaggio C

- **CARATTERISTICHE**
- linguaggio ***sequenziale, imperativo, strutturato*** a blocchi
- usabile anche come linguaggio di sistema
  adatto a software di base, sistemi operativi, compilatori, ecc.
- portabile, efficiente, sintetico
  ma a volte poco leggibile...

Linguaggio C

Basato su pochi *concetti elementari*

- dati (tipi primitivi, tipi di dato)
- espressioni
- dichiarazioni / definizioni
- funzioni
- istruzioni / blocchi

Un semplice programma

Conversione dei gradi Celsius in Fahrenheit

```
main(){
    float c, f; /* Celsius e Fahrenheit */
    printf("Inserire la temperatura da convertire");
    scanf("%f", &c);
    f = 32 + c * 9/5;
    printf("Temperatura Fahrenheit %f", f);
}
```

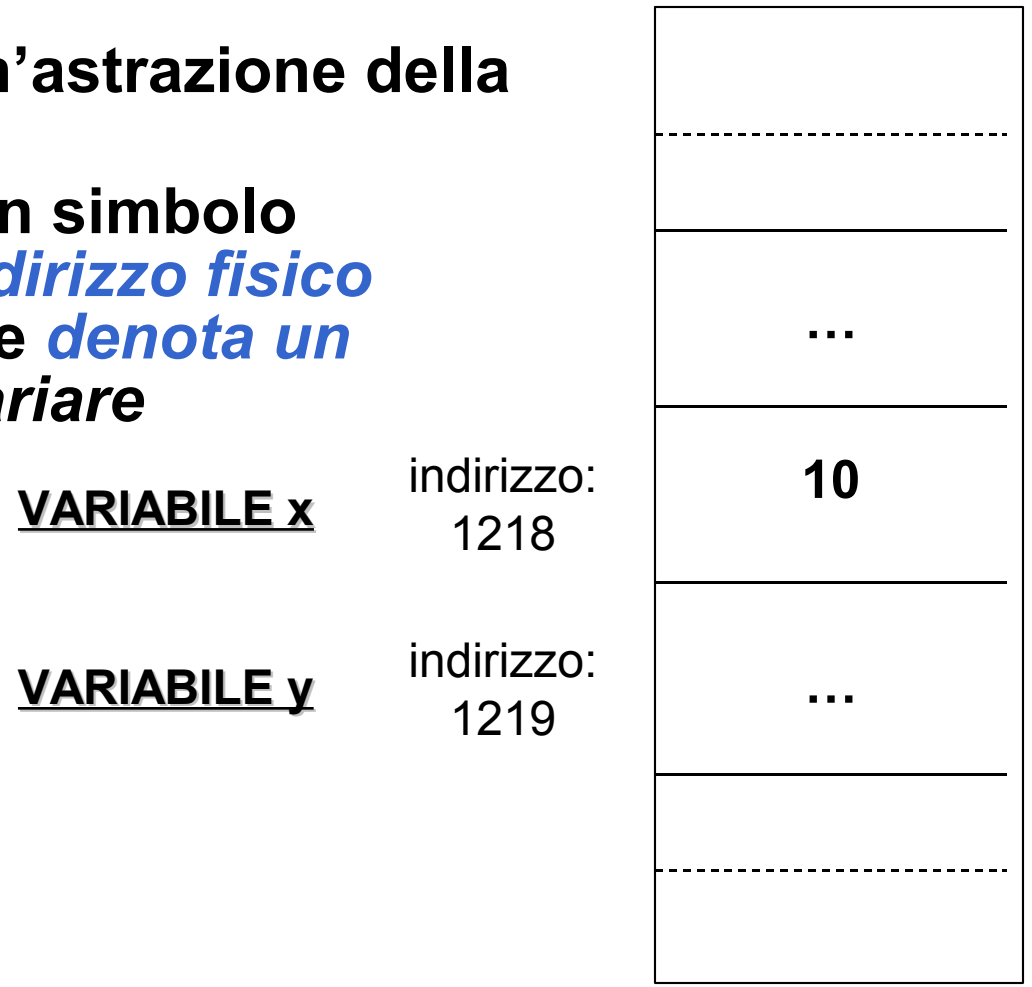
La funzione main()

La funzione ***main*** è *obbligatoria,* definita così

```
main()
{
   [<dichiarazioni-e-definizioni>]
   [<sequenza-istruzioni>]
}
```

Le variabili

- **Una *variabile* è un'astrazione della *cella di memoria***
- **Formalmente, è un simbolo *associato a un indirizzo fisico fisso (L-value)* che *denota un valore che può variare (R-value)***



- Variabile x:
  - Indirizzo 1218
  - Valore 10

Definizione di variabili

Prima di utilizzare una variabile in un programma ***deve sempre essere definita***

La **definizione** è composta da

- ***nome*** della variabile ***(identificatore)***
- ***tipo*** dei valori ***(R-value)*** che possono essere denotati alla variabile

<tipo> <identificatore>;

Esempi:

```
int x;      /* x deve denotare un valore intero */
float y;    /* y deve denotare un valore reale */
char ch;    /* ch deve denotare un carattere */
```

Inizializzazione di variabili

- Contestualmente alla *definizione* è possibile *specificare un valore iniziale* per una variabile

Definizione con inizializzazione di una variabile:

<tipo> <identificatore> = <espr> ;

Esempio:

```
int x = 32;
double velocita = 124.6;
```

- Una variabile
  - può comparire in una espressione
  - può assumere un valore dato dalla valutazione di un'espressione

Esempio:

```
double velocita = 124.6;
double tempo = 71.6;
double spazio = velocita* tempo;
```

Caratteristiche delle variabili

campo d'azione (**scope**): è la parte di programma in cui la variabile è nota e può essere manipolata, in C è determinabile ***staticamente***

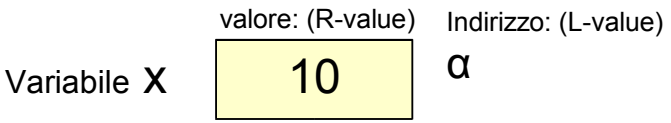
tipo: specifica la ***classe di valori*** che la variabile può assumere (e quindi gli **operatori** applicabili)

tempo di vita: è l'intervallo di tempo in cui rimane valida l'associazione simbolo/indirizzo fisico (L-value): allocazione *dinamica*

valore: è rappresentato (secondo la codifica adottata) nell'area di memoria associata alla variabile

Una *variabile* in un **linguaggio imperativo** è un'astrazione della cella di memoria associata a due diverse informazioni:

- il contenuto (R-value)
- l'indirizzo a cui si trova (L-value)



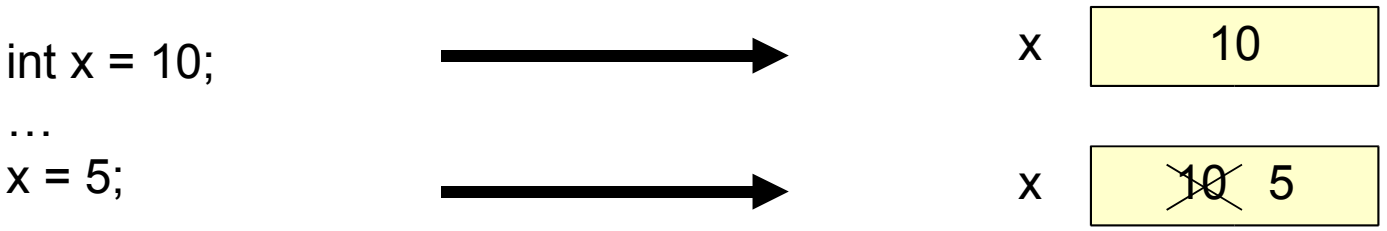
Assegnamento

- Ad una variabile può essere assegnato un valore nel corso del programma e non solo all'atto della inizializzazione

- Assegnamento di una variabile: SINTASSI

  <identificatore> = <espr>;

- L'assegnamento è l'astrazione della **modifica distruttiva del contenuto della cella di memoria** denotata dalla variabile



L'espressione *Assegnamento*

- L'assegnamento è un ***particolare tipo di espressione*** *e* come tale ***denota comunque un valore*** con un effetto collaterale: quello di ***cambiare il valore*** della variabile.

L'assegnamento è distruttivo

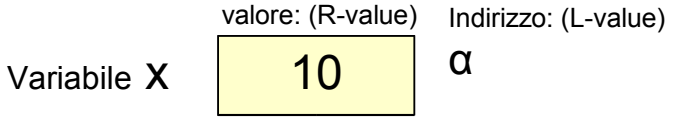
- Esempi di *espressioni di assegnamento*:

  j =0; k = j + 1;

  Se k valeva 2, l'espressione **k = j + 1;**
  - denota il valore 1 (risultato della valutazione dell'espressione)
  - *e cambia il valore di k,* che d'ora in poi vale 1 (non più 2)

Una variabile in una espressione di assegnamento:

- è interpretata come il suo R-value, *se compare a destra del simbolo =*
- è interpretata come il suo L-value, *se compare a sinistra del simbolo =*



Calcolo di un assegnamento

- Se x valeva 2, l'espressione

  x = x + 1

- denota il valore 3
- *e cambia in 3 il valore di x*
  - il simbolo x **a destra** dell'operatore = denota ***il valore attuale (R-value) di x***, cioè 2
  - il simbolo x **a sinistra** dell'operatore = denota ***la cella di memoria associata a x (L-value)*****,** a cui viene assegnato il valore dell'espressione di destra (3)
  - l'***espressione*** nel suo complesso denota il ***valore della variabile*** dopo la modifica, cioè 3

Operatori compatti di assegnamento

- Il C introduce una *forma particolare di assegnamento* che ***ingloba anche un'operazione***:

<*identif*> OP = <espressione>

è ***"equivalente"*** a

<*identif*> = <*identif*> OP < espressione>

- dove **OP** indica un operatore (ad esempio: **+**, **–**, *****, */*, …..).

- **Esempi**
- **k += j** *equivale a* **k = k + j**
- **k *= a + b** *equivale a* **k = k * (a+b)**