

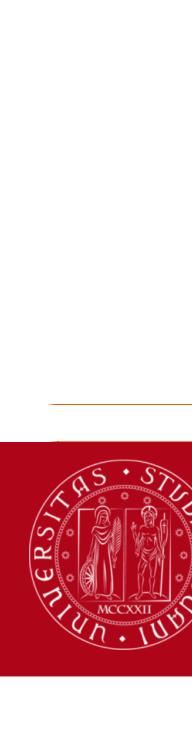
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Variabili e type safety

Stefano Ghidoni



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Agenda

- Definizioni: tipo, oggetto, valore, variabile
- Scrivere su una variabile
- Naming
- Type safety e conversioni

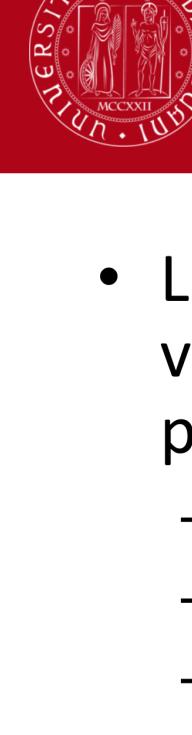
può essere anche un int, per esempio



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Tipi, oggetto, valore

- Tipo:** definisce una tipologia di contenuto, un range di valori e un insieme di operazioni per un oggetto
- Oggetto:** una regione di memoria con un tipo che specifica quale tipo di dato può essere inserito
- Valore:** l'elemento posto dentro alla variabile
 - Configurazione di bit in memoria interpretati in funzione del tipo



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Variabile

- Variabile:** un oggetto con un nome (named object)

int age = 42;

int:

age: 42

Oggetto: una regione di memoria con un tipo che specifica quale tipo di dato può essere inserito



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Tip

- Il tipo di una variabile determina le operazioni che possono essere effettuate
 - E il loro significato
- Ogni tipo ha un suo formato literal

```
39 // int
3.5 // double
'.' // char
"Annemarie" // string
true // bool
```

3.5f // float

Variabile



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Scrivere su una variabile

- L'operatore più usato è l'assegnamento (=)
- Lo stesso simbolo è usato per due operazioni simili ma concettualmente diverse
 - Inizializzazione
 - Assegnamento

int a = 3; // Qual è la differenza?

a = 4; // Qui è un inizializzatore, non un operatore

- Assegnamento composto

a += 7;

b -= 9;

c *= 2;



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Type safety

- Il tipo di una variabile condiziona:
 - La natura dei dati contenuti
 - La dimensione in byte
- Il passaggio di dati tra variabili dello stesso tipo è concettualmente semplice

Che succede se cerchiamo di trasferire dati tra variabili di tipi diversi?

- Problema del type safety

```
double d1 = 2.3;
double d2 = d1 + 2; // 2 convertito in 2.0
```



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Conversioni sicure

- Le conversioni sicure (type safe) preservano il valore (o lo convertono nella miglior approssimazione)
- Sono sicure le conversioni verso un tipo con maggior capacità
 - bool -> char
 - bool -> int
 - bool -> double
 - char -> int
 - char -> double
 - int -> double

```
double x; // valore di x indefinito
double y = x; // valore di y indefinito
```

È possibile un errore HW se x è usata senza inizializzazione

Iniziate sempre le variabili!

Il C++ è fortemente tipizzato, e la type safety deve essere sempre garantita

```
int main()
{
    int a = 20000;
    char c = a; // char non è grande abbastanza
    int b = c;
    if (a != b)
        std::cout << "Ops!\n";
    else
        std::cout << "Wow! Char è molto grande!\n";
    return 0;
}
```


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Conversioni sicure

- Le conversioni int -> double permettono di usare espressioni che contengono entrambi i tipi

```
double d1 = 2.3;
double d2 = d1 + 2; // 2 convertito in 2.0
```


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Conversioni non sicure

- Le conversioni non sicure non garantiscono che il valore convertito sia equivalente a quello di partenza ("narrowing")
 - double -> int
 - double -> char
 - double -> bool
 - int -> char
 - int -> bool
 - char -> bool

Sono raramente rilevate dal compilatore

```
int a = 20000;
char c = a; // char non è grande abbastanza
int b = c;
if (a != b)
    std::cout << "Ops!\n";
else
    std::cout << "Wow! Char è molto grande!\n";
return 0;
```


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Type safety e inizializzazione

- Osservazione: non è type safe usare una variabile non inizializzata

```
double x; // valore di x indefinito
double y = x; // valore di y indefinito
```

È possibile un errore HW se x è usata senza inizializzazione

Iniziate sempre le variabili!

Il C++ è fortemente tipizzato, e la type safety deve essere sempre garantita

```
int main()
{
    int a = 20000;
    char c = a; // char non è grande abbastanza
    int b = c;
    if (a != b)
        std::cout << "Ops!\n";
    else
        std::cout << "Wow! Char è molto grande!\n";
    return 0;
}
```


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Controllo su inizializzazione

- Le conversioni non sicure sono accettate per ragioni storiche (eredità del C)

C++ introduce una notazione di inizializzazione che evita le narrowing conversions

- { } – universal and uniform initialization

Il compilatore può controllare i literal

```
double x{2.7}; // errore: narrowing conversion
int y{x}; // errore: narrowing conversion
```

```
int a{1000}; // errore: valore fuori limiti
char b{48}; // OK
```


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Conversione di tipi

- Due modi per convertire:
 - Tipo {valore} – con controllo di narrowing
 - Tipo (valore) – senza controllo

```
double d = 2.5;
int i = d; // d2 == 1.25
int i2 = d; // i2 == 1
int i3 = {d}; // errore: double -> int, narrowing
```


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Dare nomi alle variabili

- I nomi sono importanti per gli umani!

Significativi ed evocativi

- Ma non troppo lunghi

Evitare acronimi: mtbf, TLA, myw, nbv possono confondere

– Tipico indice del loop: i (ok!)

Nomi eccessivamente lunghi sono problematici

- element_count meglio di the_number_of_elements

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Convenzione per i nomi

- È importante scegliere una convenzione e rispettarla
 - Rende molto più semplice la lettura del codice

C++ suggerita: Google C++ Style Guide

<https://google.github.io/styleguide/cppguide.html#Naming>

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Variabili e type safety

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Università degli Studi di Padova