

Tipi Predefiniti

- 2 parole sul C++
- i tipi predefiniti del C++
- Guardate il testo

il C++ non è "solo" un linguaggio,
esso riflette 40 anni di
sperimentazione e innovazione nei
Linguaggi di Programmazione (LP):

→ racchiude molti concetti che
sono il frutto di questa storia

il C++ è

possiede

possiede

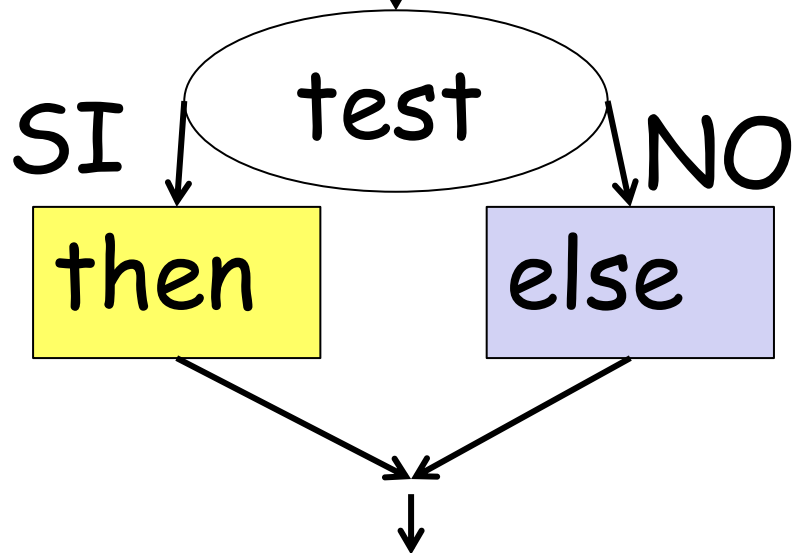
strutturato a blocchi

funzioni

tipi definiti dall'utente

un linguaggio è strutturato a blocchi

1 punto d'entrata



ed 1 punto d'uscita

TIPI

un TIPO è una coppia :

- un insieme di valori
- e da operazioni definite su questi valori

- rappresentazione sorgente (costanti)
- rappresentazione interna nella memoria (quanti byte occupano e come sono organizzati)

il C++ non prescrive quanti byte usare per realizzare i valori di ciascun tipo quindi ogni compilatore è "libero" di deciderlo

tenendo conto della CPU che deve eseguire il codice oggetto

per conoscere quanti byte vengono usati per un qualsiasi tipo T dal compilatore che si sta usando:

`sizeof(T);`

oppure se X è una variabile:

`sizeof(X);`

Tipi predefiniti

interi → **int**

reali → **float** e **double**

caratteri → **char**

booleani → **bool**

vuoto → **void**

stringhe alla C → **char***

tipo intero = int

- se abbiamo a disposizione n bit allora l'insieme di valori interi è:

$$-2^{(n-1)} \dots\dots\dots 2^{(n-1)}-1$$

- le operazioni sugli interi sono le operazioni aritmetiche e quelle di confronto

rappresentazione sorgente degli interi

- valori interi:

127

4203

-55

- devono essere tra

INT_MIN e INT_MAX

interi in base 8 e base 16

0127 rappresenta un numero in base 8

$$1*64+2*8+7 = 87 \text{ in base 10}$$

0x127 rappresenta un numero in base 16

$$1*256+2*16+7 = 295 \text{ in base 10}$$

rappresentazione interna di int

in aula informatica è presente il compilatore C++ GNU 4.4.3 esso usa per i valori interi 4 byte, cioè 32 bit

gli interi sono rappresentati in **complemento a 2** come **small-endians** che significa che i bytes sono ordinati nella memoria dal meno significativo al più significativo

operazioni frequenti:

+ - / * % (modulo)

== (uguale)

!= (diverso)

>

>=

eccetera.

i reali

- il C++ offre 2 tipi di valori reali:
float e **double**
- i double sono più precisi dei float
- le operazioni sui reali sono quelle aritmetiche e quelle di confronto

rappresentazione sorgente dei reali

12.5 12.5e2 (= 12.5 * 10²)

sono di tipo double

12.5f

è invece float

le operazioni sono + - / * == > >=
... insomma le stesse che per int

O NO ??

Rappresentazione interna: in virgola mobile (floating point)

- un bit serve per il segno, x bit per la **mantissa** e y bit per **l'esponente**
- se la mantissa rappresenta il valore m , l'esponente il valore e allora il valore reale rappresentato è **segno $1.m * 2^e$**
con segno $+$ o $-$ determinato dal bit del **segno**

nel nostro compilatore i float vengono realizzati con 4 byte mentre i double usano 8 bytes.

Per i float la mantissa col segno occupa 24 bit e l'esponente 8 bit

Torniamo al +, *, ecc. per interi e reali

sono operazioni **diverse** anche se rappresentate con gli stessi simboli

$5+2 \rightarrow +$ per interi

$2.2+3.1 \rightarrow +$ per reali

questo fenomeno si chiama

overloading (sovraccaricamento)

il compilatore usa l'operatore giusto sulla base del tipo degli operandi

il tipo `bool` ha 2 valori vero e falso
rappresentati con `true` e `false`

gli operatori principali del tipo
`bool` sono:

`!` NOT

`&&` AND

`||` OR

in C++ i 2 valori **bool** sono implementati internamente con interi:

false è rappresentato da 0

true da 1, ma anche da qualsiasi valore diverso da 0

→ occupano 1 solo byte

è un'eredità del C che non ha il tipo bool

a causa di questo fatto:

le seguenti espressioni sono
corrette in C++:

`0 && 5 == 0`

`true + false == true`

quale + viene usato ?? Lo vedremo
più avanti

i caratteri

il tipo `char` ha come dominio 256 caratteri che contengono:

- i caratteri alfabetici maiuscoli e minuscoli
- caratteri accentati
- le cifre 0,1,2,
- la punteggiatura , le parentesi
- alcuni caratteri di controllo , tra cui invio, tab

rappresentazione sorgente:

`'a' 'z' '9' ')' '.' 'è' 'ò'`

`'\n' = invio` `'\t' = tab`

rappresentazione interna dei char

occupano 1 byte

i caratteri corrispondono a interi da
-128 a 127 ASCII ESTESA

da 0 a 127 codifica ASCII

da -128 a -1 caratteri accentati e
altri

insiemi di caratteri più estesi

esistono altri tipi carattere che
comprendono un insieme di caratteri
molto più esteso (cinesi e giapponesi)

wchar_t 2 o 4 byte

UTF-8 codifica a lunghezza variabile

in ASCII esteso

i valori char sono rappresentati internamente con

'a' = 97, 'b'=98....

quindi

si può usare operazioni + e - con i valori char

ma attenzione !

'a' + 'a' = ??

tipo vuoto = void

- non ha valori
- non ha operazioni
- serve a rappresentare l'assenza di valori: al momento giusto lo useremo

stringhe di caratteri alla C

le stringhe si usano spesso per scrivere a video dei messaggi:

```
cout << "ecco un messaggio\n";
```

oppure

```
cout<<"ecco un messaggio"<<endl;
```

il tipo delle stringhe è puntatore a
carattere costante

lo capiremo più avanti

le stringhe sono rappresentate nei
programmi tra doppi apici:

"questa è una stringa"

```
#include<iostream>
using namespace std;
main()
{
    int x, y;
    cout << ``inserire 2 interi``;
    cin >> x >> y;
    cout<<``valore di x=``<< x <<``valore di
y=``<< y;
}
```

Cartella: **pluto**

File: **pippo.cpp**

comandi con cartella corrente pluto:

`g++ pippo.cpp // compila`

`./a.out // (oppure .exe) esegue`

in C++ ogni variabile ha un tipo,
perché?

- tipi diversi di valori servono per molti problemi che vogliamo risolvere
- Il tipo di un valore serve ad accedere alle sequenze di byte nella RAM nel modo appropriato
- **test di sensatezza** dei programmi: i valori sono usati in modo coerente col loro tipo