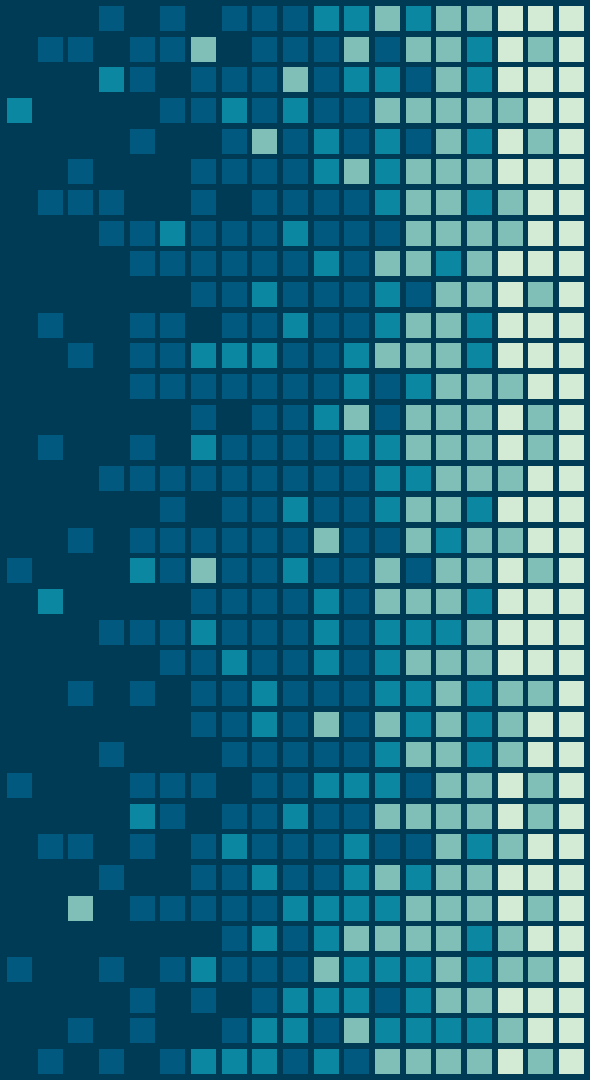
C++ from zero to hero
Parte 1



Who's who in this course?

Marcello Missiroli: prof di informatica (Corni, Unimo, unibo), nerd to the bone. Main speaker

Marco Arena: fondatore dell'Italian C++ Community, lavora in Tetra Pak, ex Ferrari. Main expert.

Luca Zomparelli: presidente di ConoscereLinux, Process Engineer in ambito Automazione. Mr Wolf.

Me & C/C++

Ho iniziato a programmare in Basic,Assembly e Pascal. Poi sono passato a C.

Ho letto il libro di Soustrup sul C++ attorno al 92.

Ho cercato di insegnare a scuola prevalentemente C puro sino al 2006, per poi passare fake C++ (C con cin e strings)

Ho ripreso il “vero” C++ solo un paio di anni fa.

Nulla di meglio di insegnare una cosa per impararla!

0.
Prologo

Che ambiente di sviluppo utilizzare?

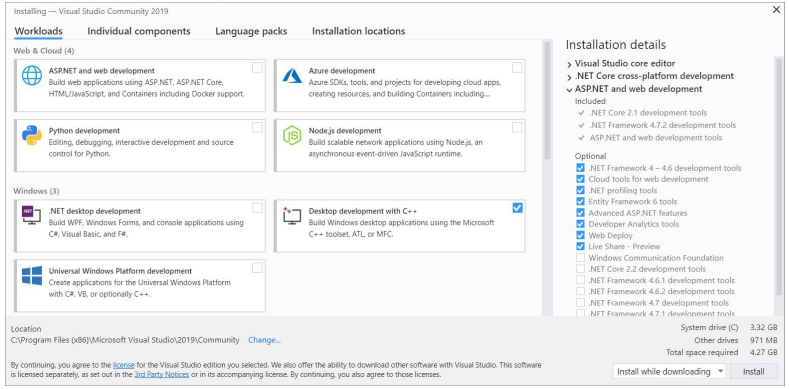
Che ambiente di sviluppo scegliere?

- Ci sono tante opzioni, scegliete ciò che ritenete più “confortevole” per voi

- Proporremo solo alcune di queste ma ricordate che Google è al vostro fianco :)
  Es: https://www.udacity.com/blog/2020/05/best-c-ides.html

#1: Windows + Visual Studio

- **Microsoft Visual Studio** è forse il miglior ambiente di sviluppo C++ in circolazione.
- Software “bello denso”
- Non lo consiglio se vi serve **solo** per questo corso
- L’edizione “Community” è gratis: visualstudio.microsoft.com



#2: Windows + Visual Studio Code

- **Microsoft Visual Studio Code** è un code editor **open source** concepito per sviluppare in maniera professionale.
- Software relativamente leggero
- Supporta un mare di estensioni (tipo Sublime Text) tra cui quella per sviluppo C++
- Si scarica da qui: https://code.visualstudio.com/
- Guida estensione C++: https://code.visualstudio.com/docs/languages/cpp

#3: Linux/MacOS + Visual Studio Code

- Le stesse cose dette prima
- Visual Studio Code è cross-platform quindi può essere utilizzato anche su Linux e MacOS
- Linux Setup: https://code.visualstudio.com/docs/setup/linux

#4: Eclipse

- L'ide Open Source più longevo. Molto diffuso in ambito education.
- https://www.eclipse.org/downloads/packages/release/luna/r/eclipse-ide-cc-developers

#5: Clion

- Un IDE molto interessante sviluppato da IntelliJ, che la fa da padrone in ambito Java e Android. Parte di una serie di Ide molto simili.
- Closed source e a pagamento, trial di 30gg.

#6: MacOs + XCode

- Un ambiente di sviluppo per MacOS molto utilizzato
- https://developer.apple.com/xcode/

#7: Compilatori online

- Per prove veloci oppure per seguire questo corso, può essere sufficiente un compilatore online
  - https://wandbox.org/
  - https://www.onlinegdb.com/ (con debugger)
  - https://godbolt.org/ (per gli audaci dell'assembly)
  - https://repl.com (shared editing, multifile, shell, git)

Note

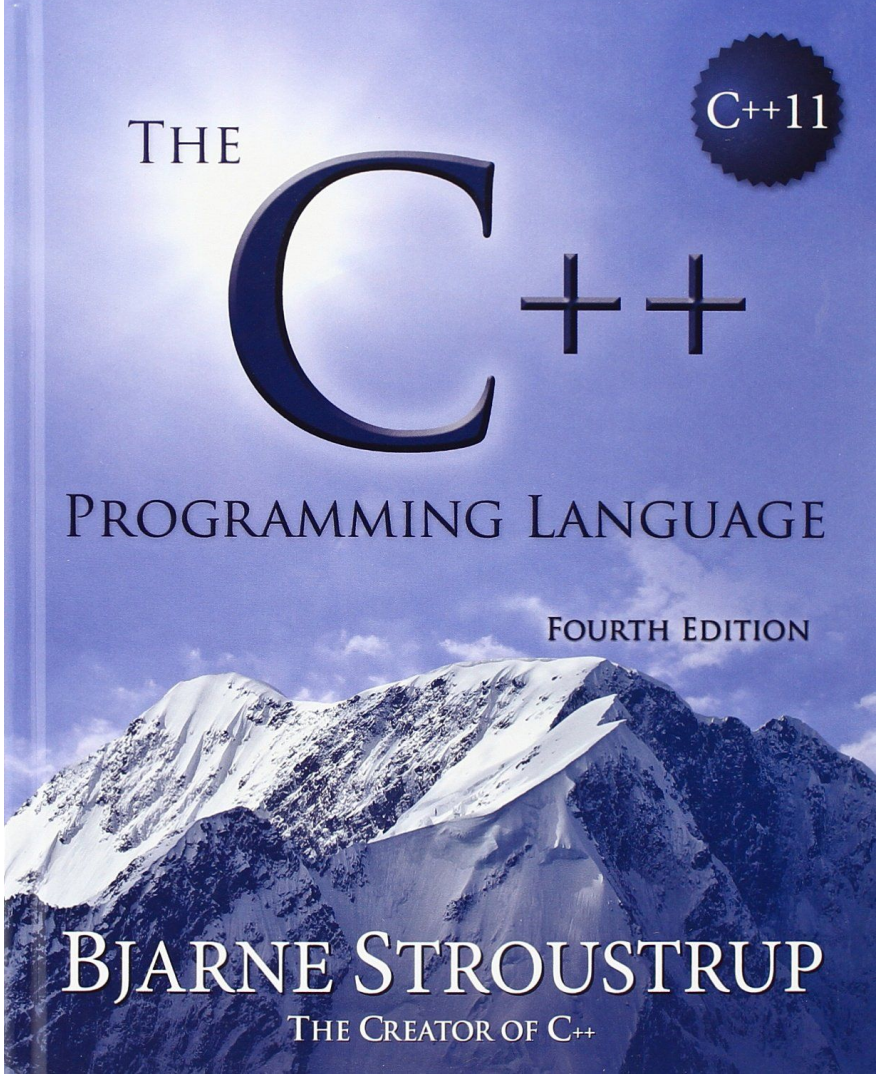
- Quasi tutti gli IDE indicati implicano l'installazione di un compilatore separato.
- Real men use **vi**

1.
Intro

C++

C++ apparso nel lontano 1983.

Pur mantenendo una buona compatibilità (90% circa) a livello di codice col C (suo predecessore), si tratta di un linguaggio diverso, soprattutto nelle sue versioni più recenti



C++

E’ un linguaggio COMPILATO, estremamente modulare. Al contrario di molti linguaggi , l’input output non fa parte del core del linguaggio.

La sintassi è basata sul C, e quindi prevedere che ogni istruzioni termini con “;” senza considerare spazi e tab.

Hello world

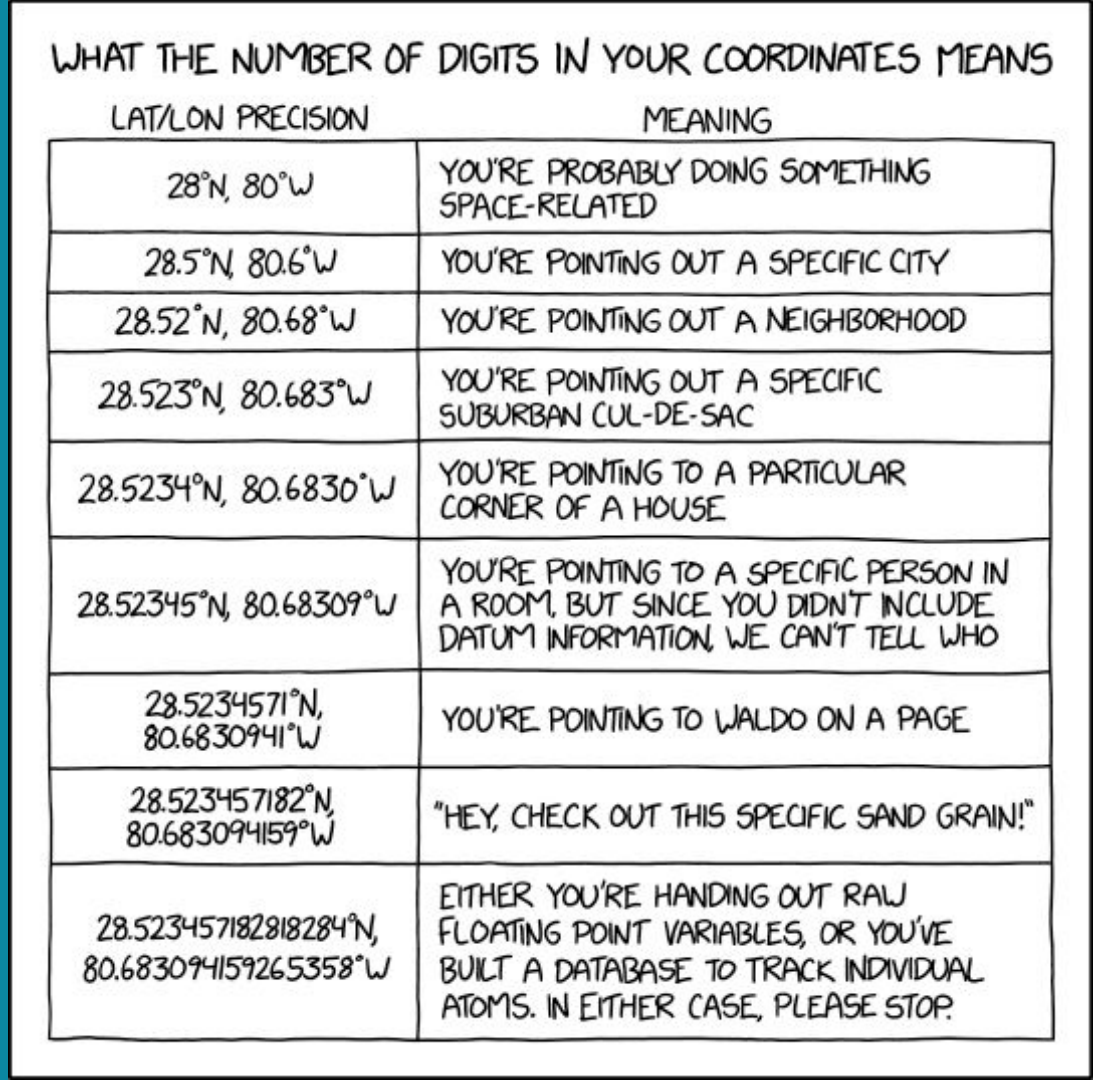
```
#include <iostream>
using namespace std;
int main()
{    cout<<"Hello World";
   return 0;
}
```

Hello world

```
#include <iostream> //libreria esterna
using namespace std;
int main() //programma principale
{   cout<<"Hello World"; //stampa
   return 0; //interazione con il SO.
}
```

2.
Le variabili e i tipi

“ *Precision counts”*



WHAT THE NUMBER OF DIGITS IN YOUR COORDINATES MEANS

LAT/LON PRECISION	MEANING
28°N, 80°W	YOU'RE PROBABLY DOING SOMETHING SPACE-RELATED
28.5°N, 80.6°W	YOU'RE POINTING OUT A SPECIFIC CITY
28.52°N, 80.68°W	YOU'RE POINTING OUT A NEIGHBORHOOD
28.523°N, 80.683°W	YOU'RE POINTING OUT A SPECIFIC SUBURBAN CUL-DE-SAC
28.5234°N, 80.6830°W	YOU'RE POINTING TO A PARTICULAR CORNER OF A HOUSE
28.52345°N, 80.68309°W	YOU'RE POINTING TO A SPECIFIC PERSON IN A ROOM. BUT SINCE YOU DIDN'T INCLUDE DATUM INFORMATION, WE CAN'T TELL WHO
28.5234571°N, 80.6830941°W	YOU'RE POINTING TO WALDO ON A PAGE
28.523457182°N, 80.683094159°W	"HEY, CHECK OUT THIS SPECIFIC SAND GRAIN!"
28.523457182818284°N, 80.683094159265358°W	EITHER YOU'RE HANDING OUT RAW FLOATING POINT VARIABLES, OR YOU'VE BUILT A DATABASE TO TRACK INDIVIDUAL ATOMS. IN EITHER CASE, PLEASE STOP.

Dichiarazione di variabili

Nella sua forma più semplice, la dichiarazione ha questa forma,

```
tipo_dato nome_variabile;
```

Se avete più variabili dello stesso tipo, potete separarle da un virgola:

```
tipo_dato nome1, nome2,
nome3;
```

Tipi di variabili scalari

- INTERI : int (4 byte), short (2 byte), long (4 byte), long long (8 byte)
- CARATTERI : char *(1 byte, NO UNICODE fino a C++11)*
- BOOLEANI : bool(1 byte*)
- VIRGOLA MOBILE : float (4byte), double (8 byte), long double (16 byte)

Nomi di variabili

- “Variabili parlanti”
- Usare variabili di una lettera solo per casi particolari
- Usare uno stile uniforme (cammello/serpente/ungherese)
- Nomi di variabili comuni: conta, temp, scambio, somma.

Definizione di costanti: due modi

```
const int TERMINE=2;
```

Dichiara una variabile che non può essere modificata.

(in C si preferiva usare un metodo basato sul preprocessore, ma decisamente più brutto e prono a errori)

Valori costanti (‘Literal’)

INTERI

Decimale: 10

Ottale: 012

Esadecimale: 0xA

Binario: 0b01010

Booleano: true

Carattere: ‘A’

VIRGOLA

Normale: 1.34

Esponenziale: 10.0E3

Inizializzazione

E’ possibile preassegnare un valore alla variabile. Esempio

int x=42;

Se non lo fate, il valore della variabile è indefinito. Le fiamme dell’inferno vi perseguiteranno, anche se può darsi che vi vada bene.

Promozione (casting automatico)

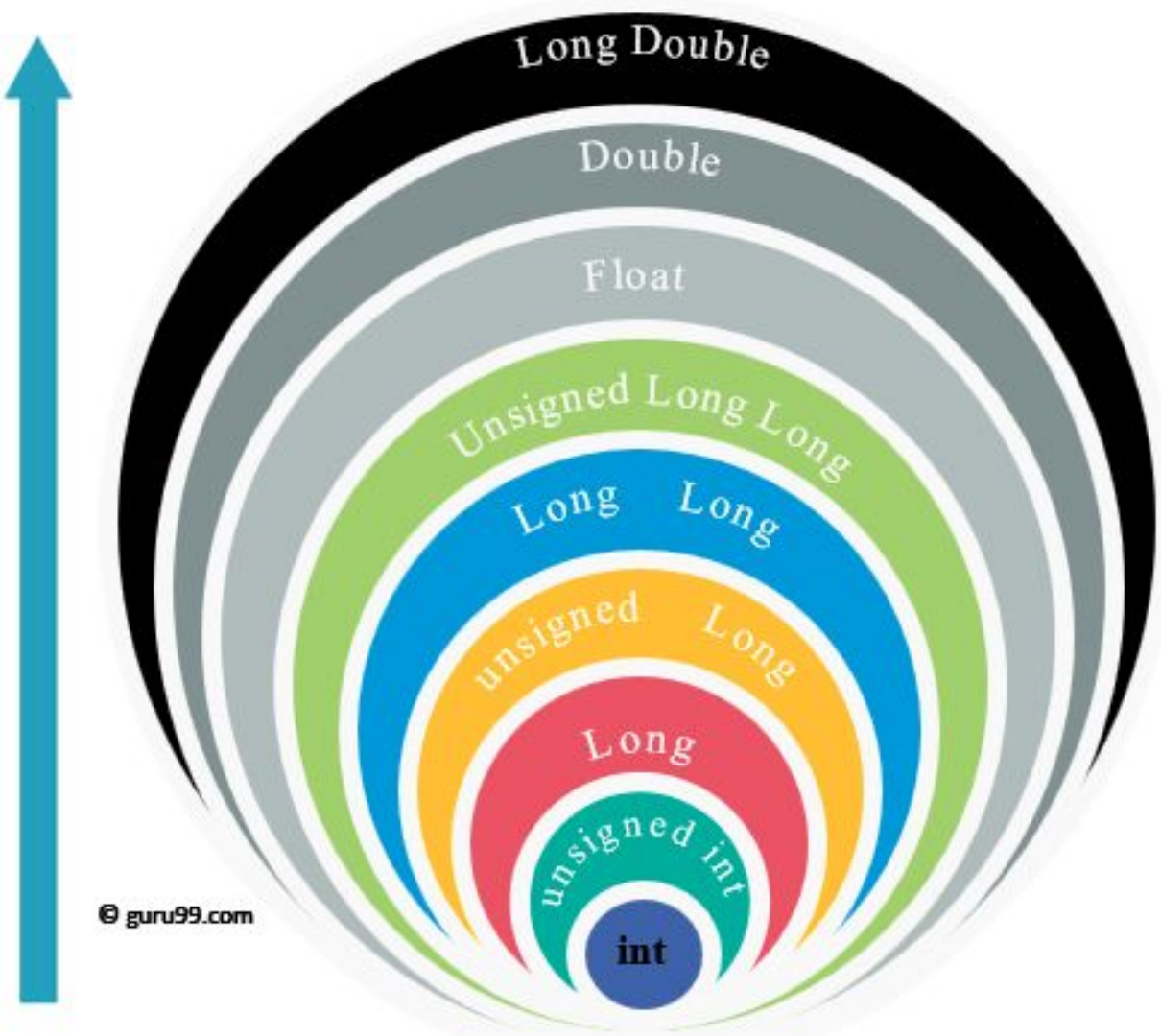
- Una variabile di un tipo può essere assegnata senza problemi a una variabile dello stesso tipo con una gamma più elevata
- int← bool
- double ← float
- int ← char (le lettere sono viste come interi)

Promozione (casting automatico)

- Lo stesso accade assegnando una variabile di tipo diverso ma compatibile. In questo caso il computer “adatta” la variabile come meglio crede.
- int ← float
- int ← double

Casting

- In altri casi il computer segnala errore. Se però volete forzare l'operazione, potete farlo col meccanismo del casting, premettendo il tipo desiderato alla variabile (o espressione)
- float x; double y=3E20;
- x = float (y);
- Potrebbe avere effetti indesiderati.


Long Double
Double
Float
Unsigned Long Long
Long Long
unsigned Long
Long
unsigned int
int
© guru99.com

Modifica di segno (tipi)

- `unsigned (sottintende int)`
- `unsigned int`
- `unsigned long int`
- `unsigned long long int`
- `unsigned char (= 1 byte)`

Modifica di tipo

- Static - variabile permanente (non si riazzera quando viene creata)
- *Extern - variabile presente in altri file (librerie, ecc)*
- *Mutable - permette di modificare variabili composte di tipo const*

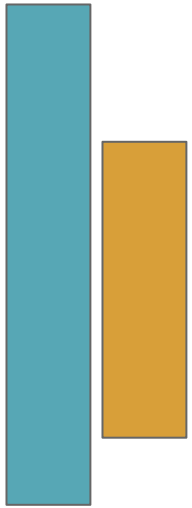
Variabili globali

- Sono definite FUORI dal main() (e come vedremo in futuro, da altre funzioni).
- Come il nome implica, sono viste e utilizzabili dal punto in cui si incontrano in poi, fino alla fine del file.
- Ne consegue che non potete avere due variabili locali con lo stesso nome.
- In termini tecnici, le variabili globali hanno uno scope di file (file scope namespace)

Variabili locali

- Le variabili locali sono definite **all'interno** di un blocco di codice, ovvero tra {}.
- Sono visibili solo all'interno del blocco.

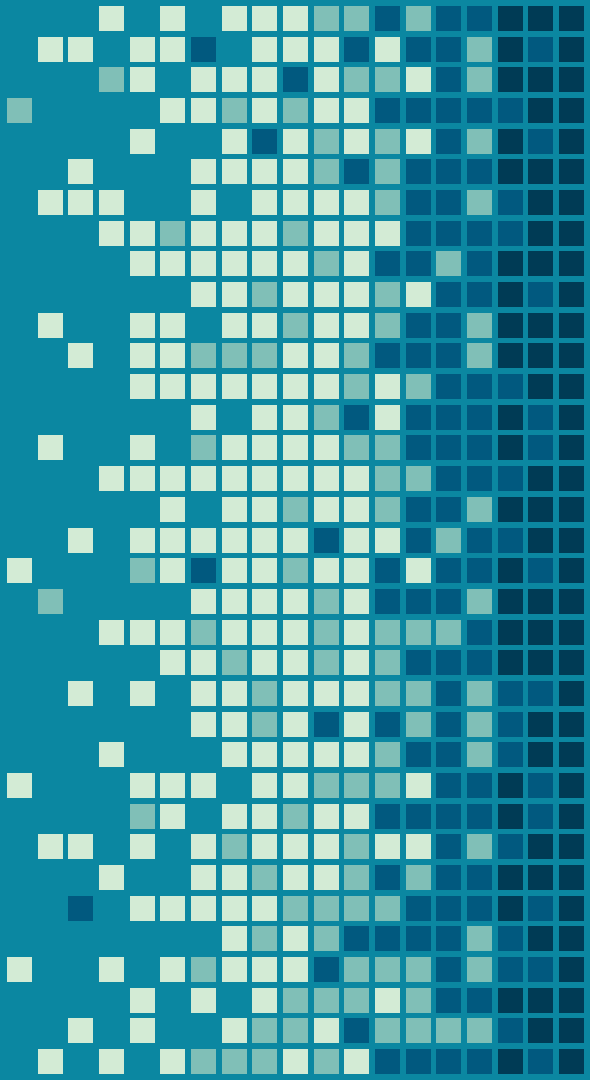
Esempio



```
int glob;
int main() {
    int loc=10;
        while (i>0)   { int temp;
                        cin>> temp;}
}
```

“Science is what we understand well enough to explain to a computer; art is everything else.”

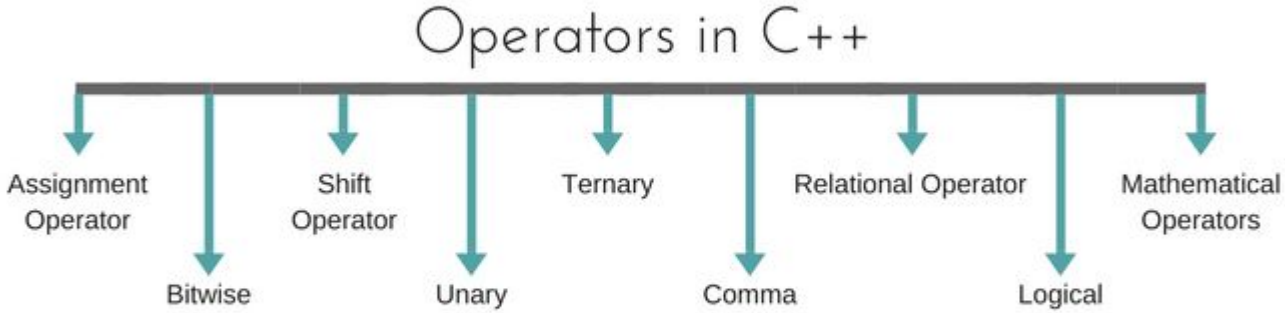
David Knuth.



3.
Operatori e operazioni

Operatori

C/C++ prevedono moltissimi operatori..



Istruzione di assegnazione

Il formato tipico è `<lvalue>=<rvalue>` o più semplicemente `<variabile>=<espressione>`

Il segno = è, in C/C++ (ma anche Java/Python/PHP…) l'operatore di assegnazione . Esempio:

```
lato=7;
```

Significa "sto assegnando il valore 7 alla variabile lato"

PRO NOTE: Quando si crea una variabile ha anche il significato di costruttore es:

int i = 0;

Operatori matematici

Binari

- +
- -
- *
- / (intera - entrambi gli operandi sono interi)
- / (reale - almeno un operando non è intero)
- % resto

Unari

- +
- –

Scorciatoie

Binari

- +=
- -=
- *=
- /=
- %=

ESEMPIO: x+=3 ⟶ x=x+3

Unari

- ++
- --

ESEMPIO:X++ ⟶ X=X+1

Incrementi unari: pitfalls

Esistono **due versioni** degli operatori di incremento unari:

PREFISSO: ++x

PRIMA incrementa la variabile, poi usa il valore modificato nell'espressione.

```
int y, x=2:
y=++x;
cout<<y;
```

POSTFISSO: x++

PRIMA usa il valore della variabile, quindi lo incrementa.

```
int y, x=2:
y=x++;
cout<<y;
```

Operatore ternario

- Riunisce assieme assegnazione e confronto.
- Efficiente e rapido, riduce la leggibilità
- Formato
  ? <condizione> : <valore vero> : <valore falso>
- Esempio
  `x= z>0 ? :4: 5`
- Equivale a `if (z>0) x=4; else x=5;`

Operatori logici e relazionali

- ==
- !=
- >=
- <=

- ! (unario)
- &&
- ||

Restituiscono sempre true (1) o false (0). Se non indicate diversamente, si applica prima il !, poi l' &&, poi gli ||. I confronti vanno sempre per ultimi

Operatori bit-a-bit

Si tratta di operatori che effettuano le operazioni di AND, OR e NOT non sul valore delle variabili, ma sulla loro rappresentazione binaria.

Sono **& (AND), | (OR) , ^ (OR esclusivo o XOR) e !**

Esempi:

X = 2 // → 00000010

Y= 11 // → 00001011

Z= X&Y //→ 00000010 → 2

K= X^Y // → 00001001 → 9

W= X|Y //→ 00001011 → 2

V= !X // → 11111101 → 253

Operatori di spostamento

Si tratta di operatori che operano sulla rappresentazione binaria del numero, “spostando” la posizione dei bit

<< sposta i bit a sinistra

>> sposta i bit a destra,

>>> sposta i bit a destra senza segno.

Nel caso di cin e cout, “sposta i dati”

Esempi:

X = 2 // → 00000010

Y= 11 // → 00001011

Z= X<<1 // → 00000100 → 4

W= Y>>2 // → 00000010 → 2

Operatore virgola

- Serve per separare elementi simili creando una lista
- Se usato in una espressione, **tutti i calcoli sono buttati via tranne l'ultimo!**
- int a,b,c=4,d;
- x= 4 , 6 , 8 → x vale 8.

4.
Funzioni

LIbrary functions (pessima traduzione)

Intro

In tutti i linguaggi di programmazione è possibile scomporre problemi complessi in moduli più semplici affrontabili singolarmente. L'elemento base di questa strategia sono le funzioni.

Sono blocchi di codice ***indipendenti*** da altri moduli, ciascuno destinato a una precisa operazione

Formato generale

```
tipo ritorno nome_funzione ([lista parametri])
{codice;}
```

Tipo ritorno può essere un qualsiasi tipo di dato standard: `int`, `char`, `long`, `unsigned`, `float`... Se non ha valore di ritorno, usare `void`.

Return

L'istruzione **`return`** indica la fine della funzione, e la restituzione di un **valore** al codice chiamante.

Il valore restituito deve essere di tipo compatibili. È ERRORE non restituire un valore, se previsto.

Il valore non deve necessariamente essere una variabile.

Se la funzione è void, si usa return senza valore.

Esempi

```
void inutile(void) {return;}
int zero() { return 0; }
void saluto() {
cout<<"Ciao!\n";
}
```

Dove collocare le funzioni?

Le funzioni devono essere dichiarate prima di essere usate, quindi tendenzialmente devono precedere il loro uso.

Tuttavia è possibile (e consigliato) separare dichiarazione della funzione (il prototipo, che deve venire prima) dalla definizione (il codice).

E' il meccanismo che permette di separare il codice in più file.

Parametri

Per passare dati alla funzione, si usano i parametri. Tipicamente si utilizza il passaggio per valore: il valore viene passato al parametro, che è considerato una variabile locale.

```
void stampa(int a) {
cout<<a;
}
```

Parametri per riferimento

Questo impedisce al programma di modificare la variabile passata per parametro. Se questo è un obbiettivo, occorre passare la variabile per riferimento. Esempio:

```
void incrementa(int &a) {a++;}
```

Esiste un terzo modo, il passaggio per indirizzo, compatibile con il C classico, un po' più complesso.

pass by reference

cup =

fillCup( )

pass by value

cup =

fillCup( )

www.mathwarehouse.com

Funzioni di libreria

- Il compilatore mette a disposizione una grande quantità di funzioni, che occorre includere se vogliamo usarle:
- `#include <cstdlib>`
- `#include <cmath>`
- `#include <ctime>`
- `#include <iostream>`

Con l'eccezione dell'ultima, si tratta di librerie C.

Funzioni matematiche

- Per la matematica è sufficiente includere math:
- `#include <cmath>`
- Il linker provvede automaticamente a inserirlo nel vostro programma.
- Troverete radici quadrate, elevamento a potenza, ecc. Ecc.

FUNZIONE MATEMATICA		FUNZIONE C++
$\sqrt{x}$	radice quadrata di *x* (restituisce un valore ≥0)	`double sqrt (double x);`
sen x	seno di *x* (con *x* reale)	`double sin (double x);`
cos x	coseno di *x* (con *x* reale)	`double cos (double x);`
tg x	tangente di *x* (con *x* reale)	`double tan (double x);`
arcsen x	arcoseno di *x* (con *x* nell'intervallo [-1 ;+1])	`double asin (double x);`
arccos x	arcoseno di *x* (con *x* nell'intervallo [-1 ;+1])	`double acos (double x);`
arctg x	arcotangente di *x* (con *x* reale)	`double atan (double x);`
e^x	*e* elevato ad *x* (con *x* reale)	`double exp (double x);`
10^x	10 elevato a *x* (con *x* reale)	`double pow10 (double x);`
ln x	logaritmo in base *e* di *x* (con *x* reale ≥0)	`double log (double x);`
log x	logaritmo di base 10 di *x* (*x* reale positivo)	`double log10 (double x);`
|x|	valore assoluto di *x*	`double fabs (double x);`
	calcolo ipotenusa di un triangolo rettangolo i di cateti *x* ed *y* (con *x*,*y* numeri reali)	`double hypot (double x, double y);`
arctg x/y	arcotangente di *y*/*x*. Il valore restituito è compreso fra $-\pi$ e $+\pi$ estremi compresi	`double atan2 (double y, double x);`
y^x	*x* elevato ad *y* (con *x*,*y* numeri reali)	`double pow (double x, double y);`
	calcola mantissa ed esponente di *x*	`double frexp (double x, int* esp);`

5.
Flusso

Teorema di Böhm-Jacopini

Il teorema che afferma che qualunque algoritmo può essere implementato in fase di programmazione utilizzando tre sole strutture controllo: la sequenza, la selezione e l'iterazione, da applicare ricorsivamente alla composizione di istruzioni elementari

(Wikipedia)

Sequenza

In C++ la sequenza è indicata da un codice racchiuso tra graffe. In generale, ogni singola istruzione può essere sostituita da una sequenza.

Inoltre le le graffe creano uno scope locale, per cui tutte le variabili dichiarate all'interno di graffe esistono solo all'interno di esse.

Scelta

La principale istruzione di scelta è l'if, con la sua estensione opzionale else.

NON mettere il ; prima dell'else.

```
if ([espressione logica])
{
    [istruzione]; ...
    }
    else {
        [istruzione]; ...
    }
```

Scelta (2)

La soluzione alternativa è lo switch con la estensione opzionale default. Ha alcune caratteristiche negative come l'obbligo delle istruzioni intere, il break; da inserire manualmente.

```
switch ([espressione intera]) {
 case [v1]:[seq. Istruzioni];
 break;
 case [v2]:[seq. Istruzioni];
 Break;
 case [v2]:case[v3]:case[v4]:
 [seq. Istruzioni];
 break;
 ...
 default:[seq. Istruzioni]
    }
```

Iterazione

Le istruzioni base sono while e do…while. Si entra e si resta nel ciclo se l'espressione è vera.

L'unica differenza è che nel secondo caso il ciclo viene sempre eseguito ALMENO una volta.

```
while ([esp. logica]) {
    [istruzione]; }

do ([esp. logica]) {
    [istruzione]; }
while ([esp. logica])
```

Iterazione (2)

Il for è una istruzione molto flessibile che in pratica racchiude tre o più operazioni.

Gli esempi sotto possono fornire qualche indizio sul loro uso.

```
for ([assegnazione];
     [esp. Logica];
     [incremento]) {
    [istruzione]; }
```

```
for (int i=0; i<10; i++);
for (; j<10; );
for (i=0,J=0; j10; i!=j);
```

Iterazione : interruzioni

E’ possibile (anche se sconsigliato) interrompere il normale flusso di un ciclo con due istruzioni;

break; interrompe il ciclo più interno e passa alle istruzioni successive.

continue; interrompe l’iterazione corrente, ed effettua un nuovo controllo.

CHALLENGE TIME!!

italiancpp.org/challenge

E’ tutto per oggi!