IL MIO PRIMO PROGRAMMA IN C++
I TIPI DI DATO E GLI OPERATORI

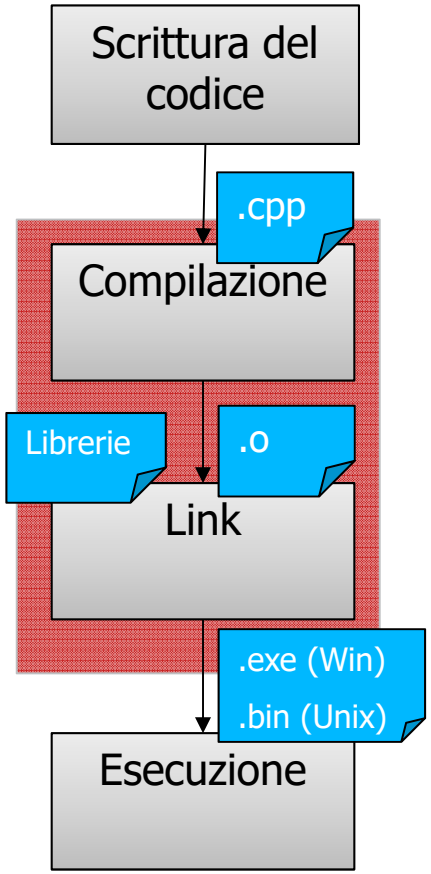
Roberto Nardone, Luigi Romano

ROADMAP

- Primo esempio di programma C++: Hello World
- Tipi di dato in C++
- Operatori

Fasi Del Ciclo di Sviluppo

- Sono riporti di seguito i passi fondamentali per ottenere un **eseguibile** da un programma C++:
  - **Scrittura del codice** – generazione di un **file sorgente .cpp**
  - **Compilazione** – compilazione del file sorgente e generazione di un **file oggetto .o**
  - **Link –** collegamento/linkaggio del file oggetto .o con le **librerie del linguaggio** e generazione dell'eseguibile **.exe (Windows) .bin (Unix)**
  - **Esecuzione** – Esecuzione del programma



CIAO MONDO


```
1   //  File: HelloWorld.cpp
2
3   #include <iostream>
4   using namespace std;
5
6
7   int main(void)
8   {
9     cout << "Hello World!" << endl;
10
11    return 0;
12  }
```

```
g++ -ohelloC hello.cpp
```

DIRETTIVE

- Alle fasi precedentemente viste ne dovremmo aggiungere un'altra, precompiling, durante la quale vengono risolte le direttive per il compilatore

- Una **direttiva** inizia sempre con il carattere **#** (a colonna 1) e occupa una sola riga (non ha un ***terminatore***)

```
#include <iostream>
```

```
#define A 3
```

- Il **preprocessore**, eseguendo le **direttive**, non produce codice binario, ma modifica il codice sorgente destinato al **compilatore**

STRUTTURA DI UN PROGRAMMA C++

- Per ogni linguaggio di programmazione (Java, Python, …) il programma più elementare è “Hello World”. In C++:

```
1   //  File: HelloWorld.cpp
2
3   #include <iostream>
4   using namespace std;
5
6
7   int main(void)
8   {
9     cout << "Hello World!" << endl;
10
11    return 0;
12  }
```

Output

> Hello World!

// indica un **commento**

La funzione **main** è la funzione **principale** che viene eseguita dal calcolatore in fase di esecuzione. E’ il cosidetto **entry point** del programma

Tutto ciò che inizia con **#** è una **direttiva** al **Preprocessore** (vedremo in seguito). Vuol dire: includi il codice standard C++ per fare operazioni di input/output (ad es. **cout**)

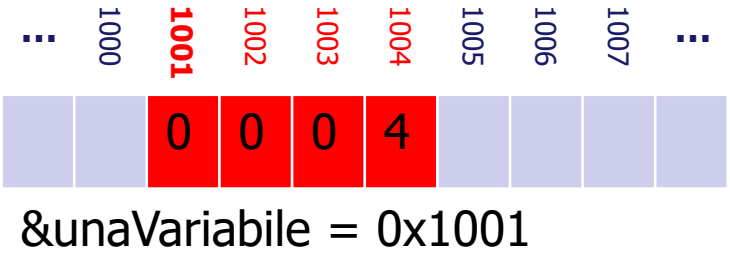
Questa linea di codice è un **Istruzione** eseguita dal calcolatore. Le **istruzioni** sono eseguite nell’ordine in cui appaiono e terminano con **';'** . **std::cout** significa: **st**andar**d** **c**haracter **out**put device. **<<** è un **Operatore**

Variabili e Tipi di Dato

LE VARIABILI C++

- In un programma C++ si fa uso di **Variabili.**

```
int unaVariabile = 4;
```



- Queste devono essere obbligatoriamente **dichiarate** prima del loro utilizzo
- Per una variabile vanno distinti tre momenti:
  - **Dichiarazione:** il programma dichiara l'esistenza della variabile, il suo tipo ed il suo nome, ma non viene realmente allocata memoria
  - **Definizione:** la variabile viene associate ad uno spazio di memoria (viene attribuito un indirizzo)
  - **Inizializzazione**: alla variabile viene associato un valore
- Spesso **Dichiarazione** e **Definizione** avvengono nello stesso istante

DEFINIZIONE E DICHIARAZIONE

- Una variabile può essere di diversi **tipo**. I più elementari sono:
  - **Intero** (*int*)
  - **Booleano** (*bool*): (Vero o Falso)
  - **Carattere** (*char*)
  - **Floating-point** (*float, double*) (numeri razionali)

```
#include <iostream>

int main() {
      int prima_variable;
      prima_variabile=5;
}
```

- **Non tutti i tipi occupano la stessa quantità di memoria**

`sizeof()` ritorna la dimensione del dato passato come argomento

Es. `sizeof(int)` ritorna 4

INFORMAZIONE

- Il concetto di *informazione* può essere definito in modo semplicistico come la risoluzione di un'incertezza, la scelta tra più alternative

- Ogni *informazione* è caratterizzata da *tipo*, *valore* e *attributo*
  - Il *tipo* specifica a quale insieme di enti appartieni l'informazione
    - numero intero, numero reale, nome di persona, lettera dell'alfabeto, data,…
  - Il *valore* è un elemento specifico di un insieme
    - 5, 5.3, Mario, c, 03/03/2008,…
  - L'*attributo* specifica ulteriormente l'informazione e ne dà un senso compiuto
    - l'informazione di tipo *numero* e valore *1000* potrà avere come attributi "costo della rata mensile", "numero di telaio",…

TIPI

- In un programma C++ tutti i nomi delle entità *(identificatori)* utilizzati devono essere *dichiarati*, ossia deve essere specificato il loro *tipo*

- Ogni *tipo* determina quali operazioni possono essere applicate ad un nome e come tali operazioni sono interpretate

```
float x;        // x is a floating-point variable
int y = 7;      // y is an integer variable with the initial value 7
float f(int);   // f is a function taking an argument of type int and returning a floating-point number

x = y + f(2);
```

TIPI FONDAMENTALI

- Il tipo **booleano** può assumere due valori: Vero (**true**) o Falso (**false**). Serve per effettuare **operazioni logiche**

```
#include <iostream>
int main() {
      bool var;
      var=true;
}
```

```
#include <iostream>
int main() {
      bool var= true;
}
```

- Il tipo **Float** è utilizzato per operazioni tra numeri reali

```
#include <iostream>
int main() {
      float variabile_1, variabile_2;
      variabile_1=4.5;
      variabile_2=5.5;
}
```

- Il tipo **Char** indica un carattere

```
#include <iostream>
int main() {
      char carattere='a';
}
```

ATTENZIONE, NOTARE CHE…

- Il C++ è **Case-Sensitive**: vuol dire che fa differenza tra maiuscole e minuscole
  - Ad es. `float variabile_1;` ≠ `float Variabile_1;`

- Non è possibile definire variabili con spazi. Per convenzione lo spazio è inserito con tratto basso _
  - Ad es. `int prima_variabile;`

- Le parole chiave (quelle evidenziate in grassetto) sono riservate del linguaggio. Non si possono usare per altri scopi. Il compilatore darà errore!!

DIETRO LE QUINTE

- I valori delle variabili di un programma vengono registrati nella **memoria del calcolatore**
- Il calcolatore deve quindi conoscere quali valori vogliamo memorizzare in una variabile in quanto deve sapere quanta memoria riservare per tali valori
- **Non tutti i tipi di valore occupano la stessa quantità di memoria**
- Quantità di memoria diversa per registrare un carattere, un piccolo numero o un grande numero
- La memoria di un calcolatore è suddivisa in **byte (8 bit)**.
- A seconda del tipo, una variabile sarà *rappresentabile* su *n* bit

RAPPRESENTAZIONE DEGLI INTERI – 1/2

- Il C++ permette di definire variabili intere di tipo **signed** (con segno) o **unsigned** (senza segno). Di *default* sono definite come **signed**. Ad es. quando scriviamo:
  - **int** variabile_intera=-30**;**
- *variabile_intera* sarà **con segno**.
- Anteponendo la parola chiave **unsigned**, è possibile anche definirle come variabili senza segno. Ad es:
  - **unsigned int** variabile_senza_segno=24**;**
- Usando **unsigned**, potremo rappresentare numeri positivi più grandi. Ad es. nel caso degli interi **int** (n=32b), da 0 a 4.294.967.295.
- Usando **signed**, invece, i numeri positivi potranno raggiungere un valore massimo di 2.147.483.647

RAPPRESENTAZIONE DEGLI INTERI – 2/2

- Il linguaggio permette inoltre di definire diverse tipologie di interi che saranno rappresentati su *n* differenti valori di bit e dunque caratterizzati da un differente **dominio (o range)**
- In generale, il **dominio** è definito come segue:

$$\begin{cases} [-2^{n-1}+1, 2^{n-1}-1];\ signed \\ [0, 2^{n}-1];\ unsigned \end{cases}$$

  - **short int** → n=16bit → $\begin{cases} [-32769,32767];\ signed \\ [0,65535];\ unsigned \end{cases}$
  - **int** → n=32bit. In verità, la sua lunghezza dipende dal sistema operativo e dal calcolatore. Ad esempio, in MSDOS è di 16 bit mentre in sistemi a 32 bit (quali Windows 9x/2000/NT) è di 32 bit (4 bytes).
  - **long** → n=32bit
  - **long long** → n=64bit

RAPPRESENTAZIONE DI BOOL

- Il **bool** come detto può assumere solo due valori **true** e **false**
- Dunque, è rappresentabile attraverso un solo bit
- Conversione **bool** → **int**:

  int valore_esempio= **true**;

  - valore_esempio assume valore 1

  int valore_esempio = **false**;

  - valore_esempio assume valore 0
- Conversione **int** → **bool**:

  bool val = -100;

  - val assume valore **true**. In generale, tutti i valori diversi da zero assumono valore **true**. Solo se 0 assume valore **false**:

  bool stop = 0;

RAPPRESENTAZIONE DEI CHAR

- La rappresentazione dei caratteri segue il codice ASCII (8 bit)
- Esistono tre tipi: **char**, **signed char** e **unsigned char**
- La rappresentazione (con o senza segno) dei **char** è lasciata libera dallo standard
- Dominio char: $\begin{cases} [-128,127\,]; \ signed \\ [0,255\,]; \ unsigned \end{cases}$

ESEMPIO CODICE ASCII - CARATTERI STAMPABILI

Binario	Oct	Dec	Hex	Glifo	Binario	Oct	Dec	Hex	Glifo	Binario	Oct	Dec	Hex	Glifo
010 0000	040	32	20	Spazio	100 0000	100	64	40	@	110 0000	140	96	60	`
010 0001	041	33	21	!	100 0001	101	65	41	A	110 0001	141	97	61	a
010 0010	042	34	22	"	100 0010	102	66	42	B	110 0010	142	98	62	b
010 0011	043	35	23	#	100 0011	103	67	43	C	110 0011	143	99	63	c
010 0100	044	36	24	$	100 0100	104	68	44	D	110 0100	144	100	64	d
010 0101	045	37	25	%	100 0101	105	69	45	E	110 0101	145	101	65	e
010 0110	046	38	26	&	100 0110	106	70	46	F	110 0110	146	102	66	f
010 0111	047	39	27	'	100 0111	107	71	47	G	110 0111	147	103	67	g
010 1000	050	40	28	(	100 1000	110	72	48	H	110 1000	150	104	68	h
010 1001	051	41	29	)	100 1001	111	73	49	I	110 1001	151	105	69	i
010 1010	052	42	2A	*	100 1010	112	74	4A	J	110 1010	152	106	6A	j
010 1011	053	43	2B	+	100 1011	113	75	4B	K	110 1011	153	107	6B	k
010 1100	054	44	2C	,	100 1100	114	76	4C	L	110 1100	154	108	6C	l
010 1101	055	45	2D	-	100 1101	115	77	4D	M	110 1101	155	109	6D	m
010 1110	056	46	2E	.	100 1110	116	78	4E	N	110 1110	156	110	6E	n
010 1111	057	47	2F	/	100 1111	117	79	4F	O	110 1111	157	111	6F	o
011 0000	060	48	30	0	101 0000	120	80	50	P	111 0000	160	112	70	p
011 0001	061	49	31	1	101 0001	121	81	51	Q	111 0001	161	113	71	q
011 0010	062	50	32	2	101 0010	122	82	52	R	111 0010	162	114	72	r
011 0011	063	51	33	3	101 0011	123	83	53	S	111 0011	163	115	73	s
011 0100	064	52	34	4	101 0100	124	84	54	T	111 0100	164	116	74	t
011 0101	065	53	35	5	101 0101	125	85	55	U	111 0101	165	117	75	u
011 0110	066	54	36	6	101 0110	126	86	56	V	111 0110	166	118	76	v
011 0111	067	55	37	7	101 0111	127	87	57	W	111 0111	167	119	77	w
011 1000	070	56	38	8	101 1000	130	88	58	X	111 1000	170	120	78	x
011 1001	071	57	39	9	101 1001	131	89	59	Y	111 1001	171	121	79	y
011 1010	072	58	3A	:	101 1010	132	90	5A	Z	111 1010	172	122	7A	z
011 1011	073	59	3B	;	101 1011	133	91	5B	[	111 1011	173	123	7B	{
011 1100	074	60	3C	<	101 1100	134	92	5C	\	111 1100	174	124	7C	|
011 1101	075	61	3D	=	101 1101	135	93	5D	]	111 1101	175	125	7D	}
011 1110	076	62	3E	>	101 1110	136	94	5E	^	111 1110	176	126	7E	~
011 1111	077	63	3F	?	101 1111	137	95	5F	_					

RAPPRESENTAZIONE DEI TIPI A VIRGOLA MOBILE

- I numeri reali sono di tre tipi:
  - **float**: singola precisione (n=32 bit)
  - **double**: doppia precisione (n=64 bit)
  - **long double**: doppia precisione estesa (n=80 bit)
- I tipi a virgola mobile sono rappresentati attraverso la **notazione esponenziale** (es. +5.37E+16)

ESERCIZIO - ECHO

- Scrivere un programma che, dopo aver chiesto all'utente di inserire un carattere, lo stampa sul terminale

- Scrivere un programma che, dopo aver chiesto all'utente di inserire un numero intero, lo stampa sul terminale

DIMENSIONI

- Alcune proprietà del C++ non sono specificate nello standard e quindi dipendono dalla particolare implementazione del linguaggio di programmazione
  - Ad esempio, la dimensione del tipo int non è specificata dallo standard e può variare da compilatore a compilatore

- Affinché il codice sia portabile (ossia che funzioni su ogni sistema) è necessario che un programmatore eviti tali dipendenze

DIMENSIONI E DOMINIO DELLE VARIABILI –2/2

Tipo	Byte	Descrizione	Dominio
char	1	carattere o intero di 8 bit.	**signed:** -128 ... 127 **unsigned:** 0 ... 255
short	2	intero di 16 bit.	**signed:** -32768 ... 32767 **unsigned:** 0 ... 65535
long	4	intero di 32 bit.	**signed:**-2147483648 ... 2147483647 **unsigned:** 0 ... 4294967295
int	*	Intero. La sua lunghezza dipende dalla lunghezza del tipo **word** usato dal sistema operativo. Ad esempio, in MSDOS è di 16 bit mentre in sistemi a 32 bit (quali Windows 9x/2000/NT) è di 32 bit (4 bytes).	See **short**, **long**
float	4	numero in virgola mobile.	3.4e + / - 38 (7 cifre decimali)
double	8	numero in virgola mobile in doppia precisione.	1.7e + / - 308 (15 cifre decimali)
long double	10	numero in virgola mobile in doppia precisione estesa.	1.2e + / - 4932 (19 cifre decimali)
bool	1	Valori Booleani. Può assumere uno dei due valori: true o false.	**true** or **false**

TIPO VOID

- Il tipo ***void*** sintatticamente è un tipo fondamentale
  - Può essere usato solo come parte di un tipo più complesso, dato che non esistono oggetti di tipo ***void***

- È usato per specificare che una funzione non ritorna un valore oppure come tipo base per puntatori a oggetti di tipo sconosciuto

```
void x;      // error: there are no void objects
void f();    // function f does not return a value
void* pv;    // pointer to object of unknown type
```

IL QUALIFICATORE CONST

- Può essere anteposto alla definizione della variabile
  - `const int a = 3;`

- Sta ad indicare che il valore della variabile a non potrà modificarsi nel tempo
  - a ++; // errore!!!
  - a = a + 7; // errore!!!