

Programmazione

2 crediti

restanti 8 crediti al II semestre

il testo è:

Programmazione consapevole
di G.Filè,
nelle librerie Progetto

informazioni sul C++ sulla rete:
www.cplusplus.com

sito del corso è:

<https://elearning.studenti.math.unipd.it/labs/>

→ci troverete TUTTO

→FORUM in cui fare e rispondere a domande

per la prossima settimana:

- 13/10 11:30-13:30 lum 250

-14/10 14:30-16:30 lab. 140 (uso PC)

poi lo metterò sull'elearning

-compitino nella settimana del 16/11

-vale per il voto finale

- faremo 5 compitini durante l'intero corso
- fatti in laboratorio
- corretti automaticamente
- conteranno per il 15% del voto finale
- ogni compitino conta 3%

- partecipazione volontaria

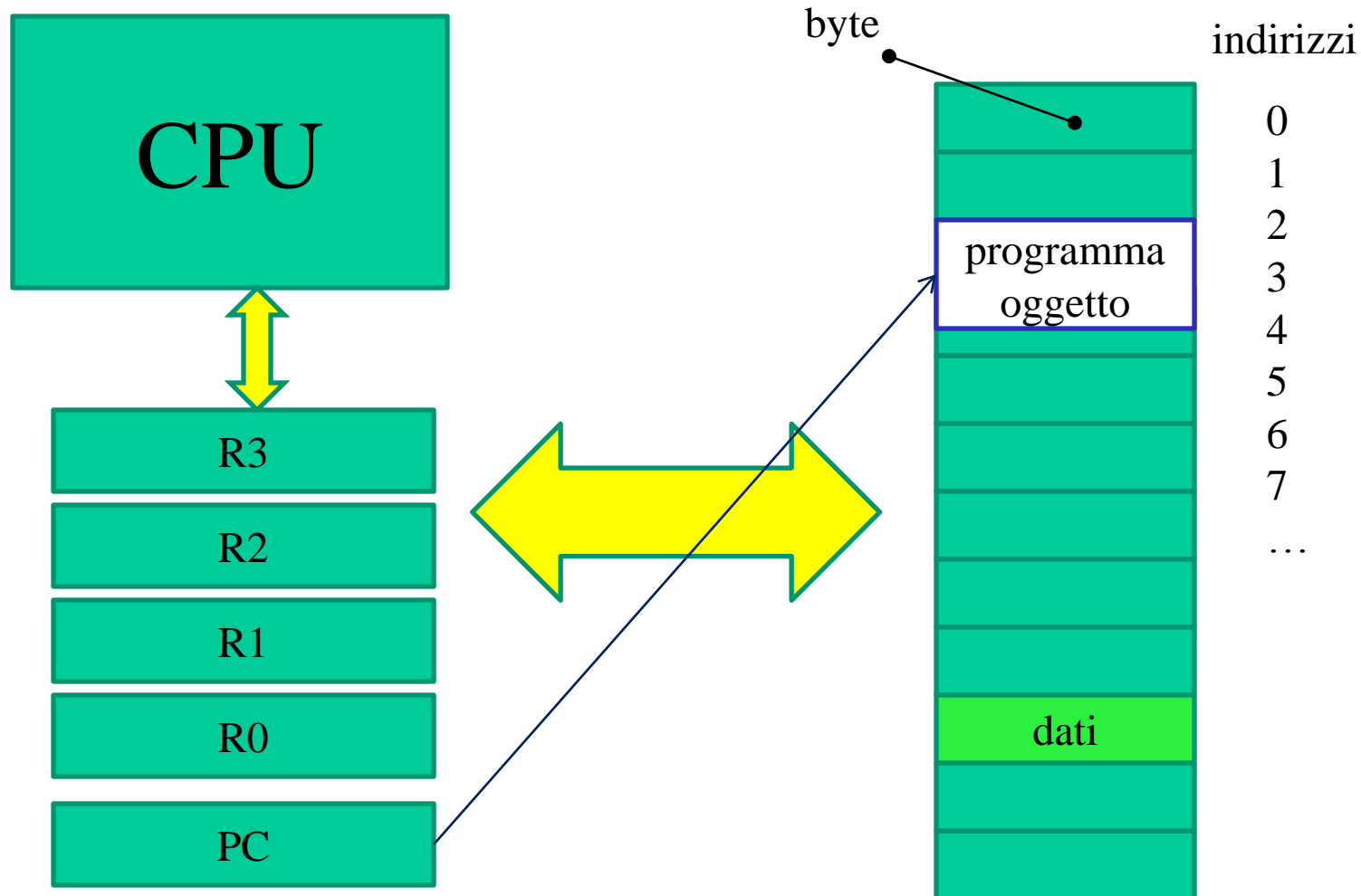
- 5 esami scritti (luglio, settembre e gennaio 2017)

- tanti esercizi

Cosa faremo

- architettura di von Neumann
- Linguaggio C++ minimale (cap. 2, 3 e 5.2 del testo)
- qualche programma con dimostrazioni di correttezza (cap. 4 del testo)
- uso del laboratorio per “far girare” gli esercizi
- compitini ed esami in laboratorio

architettura di von Neumann





il **compilatore** C++
(che è un programma)
traduce il programma
C++ (**sorgente**)
nel programma
eseguibile (**oggetto**)

come compilare ed eseguire un programma:

scrivere un programma su un file, per
esempio `prova.cpp`

`g++ prova.cpp` lo compila in `./a.exe (.out)`

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

Il C(++) in una nocciolina

- dichiarazioni
- input/output
- assegnazione
- condizionale
- while

Dichiarazioni (cap 2)

Nei programmi usiamo variabili.

Solamente di tipo intero e booleano
una dichiarazione è:

int x=0, y=1, z=2, w;

tipo intero x, y, z sono dichiarate con
inizializzazione, mentre w non è inizializzata
quindi è indefinita: vietato usarne il valore

```
int x=1;
```

x ha valore 1

e questo 1 è in memoria a qualche indirizzo

1 è l'**R-valore** di x e l'indirizzo è l'**L-valore** di x

i valori interi rappresentabili nel computer sono un intervallo:

lim_inf.....0.....lim_sup

INT_MIN

INT_MAX

sono costanti che il C++ mette a disposizione

#include<climits> o <limits.h>

Variabili booleane

`bool x, ultimo= false, primo=true;`

solo 2 valori di tipo bool: true e false

input/output (cap 3.1)

standard input = tastiera **cin**
standard output = video **cout**

```
int x=0, y=3;
```

```
cout<< x << y; // stampiamo 0 3  
cin >> x >> y; //leggiamo nuovi  
              //valori
```

il programma più semplice che c'è

```
#include<iostream> // preprocessing  
using namespace std;
```

```
main()  
{  
int x = 0, y=3;  
cout << x << y;  
cin  >> x >> y;  
}
```


assegnazione (cap 3.2)

```
int x=0, y=2;
```

```
x = y * 12 ;
```

2*12 viene assegnato a x

```
y=x-1;
```

```
x=y*x;
```

$x = y * x;$

L-valore
di x

R-valori

perché è lì
che si deve
mettere il
valore di $y * x$

per fare la
somma

condizionale o if-then-else

```
if (esp_bool)
```

```
{
```

```
    C1
```

```
    blocco then
```

```
}
```

```
else
```

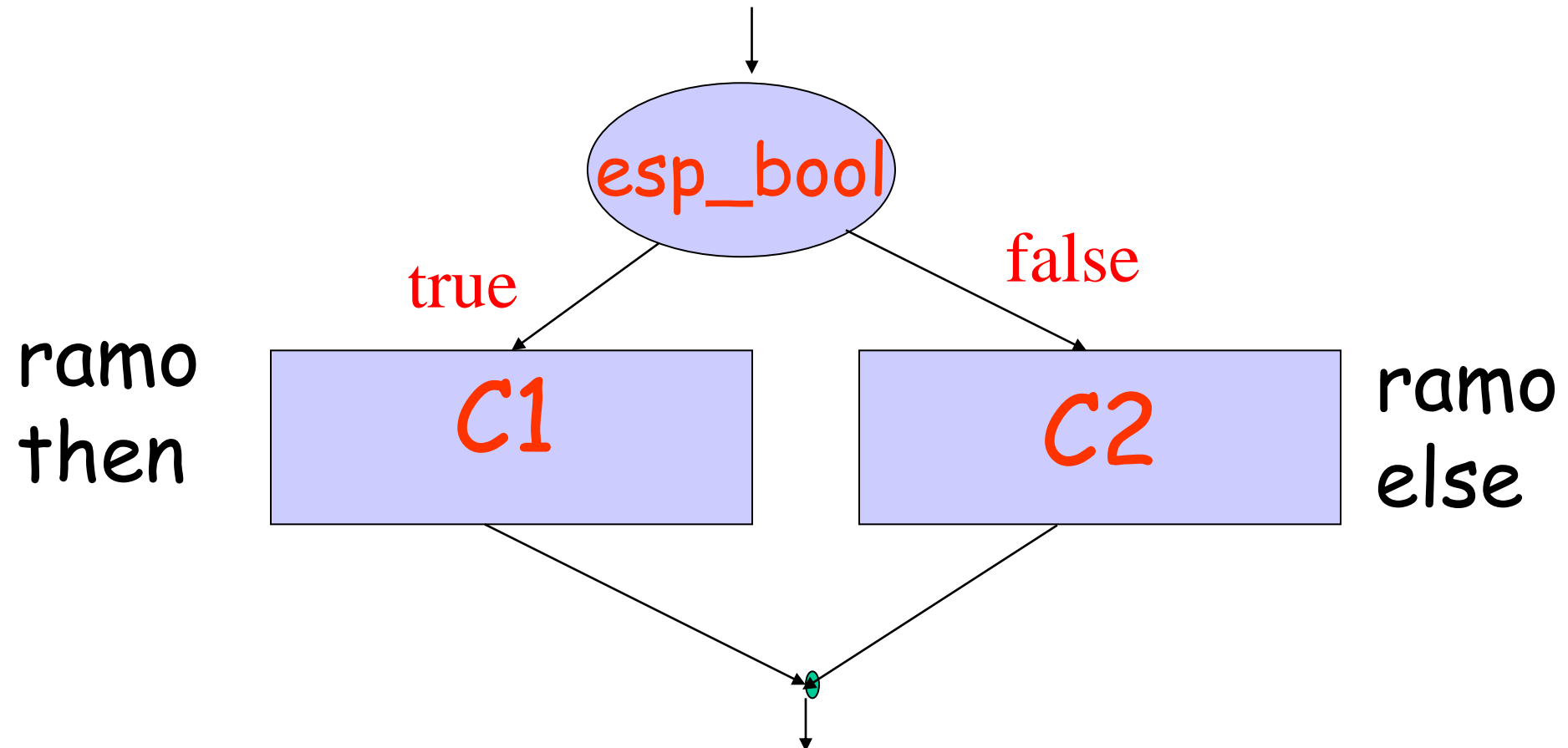
```
{
```

```
    C2
```

```
    blocco else
```

```
}
```

diagramma di flusso



1 punto d'entrata ed 1 d'uscita

//PRE=(cin contiene v1 e v2)

```
main()  
{int x, y;  
cout<<"inserire 2 interi";  
cin >> x >> y;  
cout<<" valore di x="<<x<<"valore  
di y="<<y;  
int SOM=x+y;
```

//POST=(x ha R-valore v1, y ha R-valore v2
e SOM ha R-valore v1+v2)

//POST=(x ha R-valore v1, y ha R-valore v2 e SOM ha R-valore v1+v2)

```
if(SOM>0)
    y=SOM;
else
    x=SOM;
}
```

//POST1=(SOM>0 => y=SOM && x=v1)
&& (SOM <=0 => x=SOM && y=v2)

Non è sempre vero che la post dopo un condizionale sia un condizionale

```
if(x < y)
  SOM=x;
else
  SOM=y;
```

//POST=(SOM è il valore minore tra x e y)

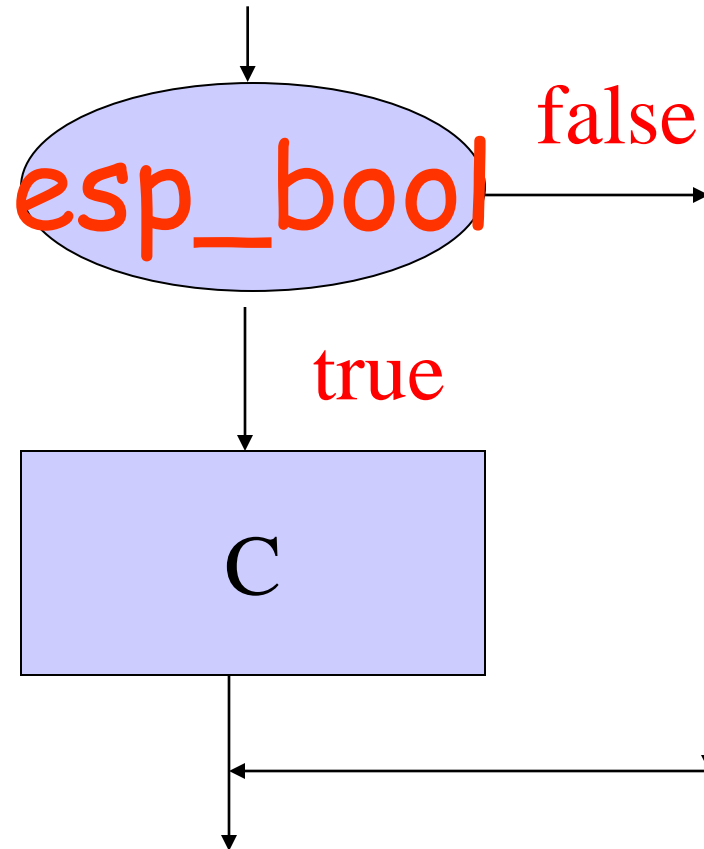
ma anche questa sarebbe possibile:

$(x < y \Rightarrow \text{SOM} = x) \ \&\& \ (x \geq y \Rightarrow \text{SOM} = y)$

[http://www.math.unipd.it/userlist/subscribe/
?idlist=199](http://www.math.unipd.it/userlist/subscribe/?idlist=199)

if(esp_bool) {C}

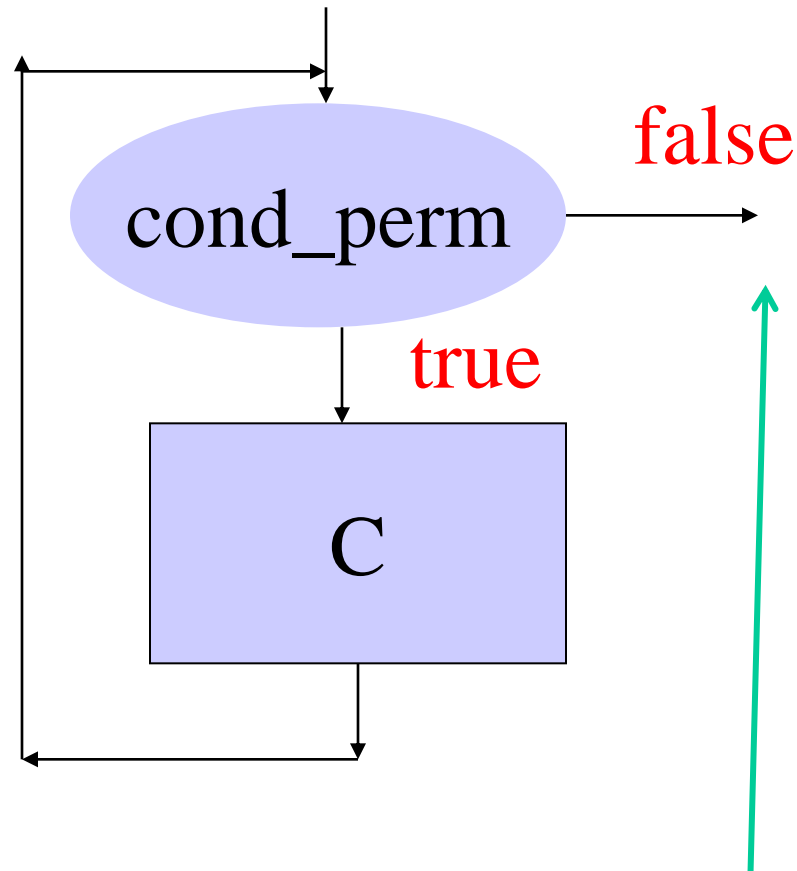
solo il
ramo then



comando iterativo o while

```
while(cond_perm)
{
    C    corpo del while
}
```

while



1 punto d'entrata ed 1 d'uscita

esempio di while:

```
int x=0;
```

```
while(x < 10)
```

```
{
```

```
    x=x+1;
```

```
}
```

```
//POST= (x=10)
```

```
int x=0;
```

```
while(x < 10)
```

```
R={0<= x <=10}
```

```
{
```

```
{0<= x <10}
```

```
    x=x+1;
```

```
}
```

```
uscita R && { x >=10}
```

```
//POST=(x=10)
```

valgono 3 fatti:

- 1) R è vera la prima volta che l'esecuzione arriva al ciclo (condiz. iniziale)
- 2) R vale ogni volta che l'esecuzione ritorna all'inizio del ciclo (invarianza)
- 3) $R \ \&\& \ !(\text{cond_perm}) \Rightarrow \text{POST}$ (condiz. d'uscita)

che succede in questo caso?

```
int x=3;
```

```
while(x < 10)
```

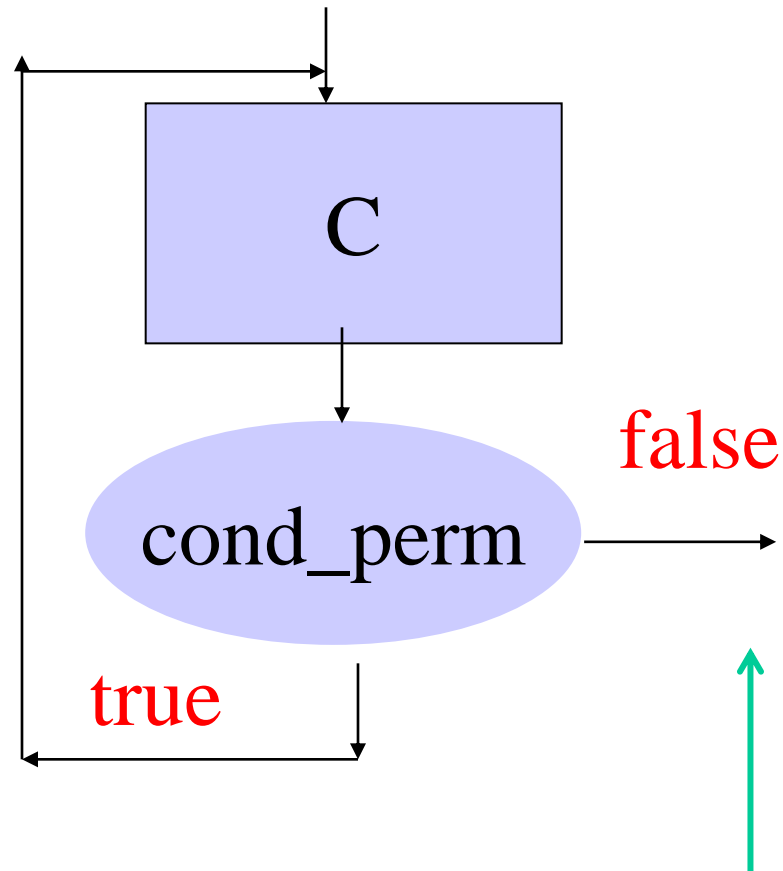
```
{
```

```
    x=x+1;
```

```
}
```

```
//POST= (x=10)
```

do-while



1 punto d'entrata ed 1 d'uscita


```
int x=0;
```

```
do      {0<= x <10}
```

```
    x=x+1;
```

```
while(x < 10); R={1<= x <=10}
```

```
uscita  R && { x >=10}
```

```
//POST=(x=10)
```