Programmazione

fatti 2 crediti

restanti 8 crediti adesso

Il testo è:

Programmazione consapevole

di G. Filè,

nelle librerie Progetto

informazioni sul C++ sulla rete:

- www.cplusplus.com

il moodle del corso 17/18 è:

http://elearning.studenti.math.unipd.it/labs/

→ci troverete TUTTO

→FORUM in cui fare e rispondere a domande

sequenza delle lezioni

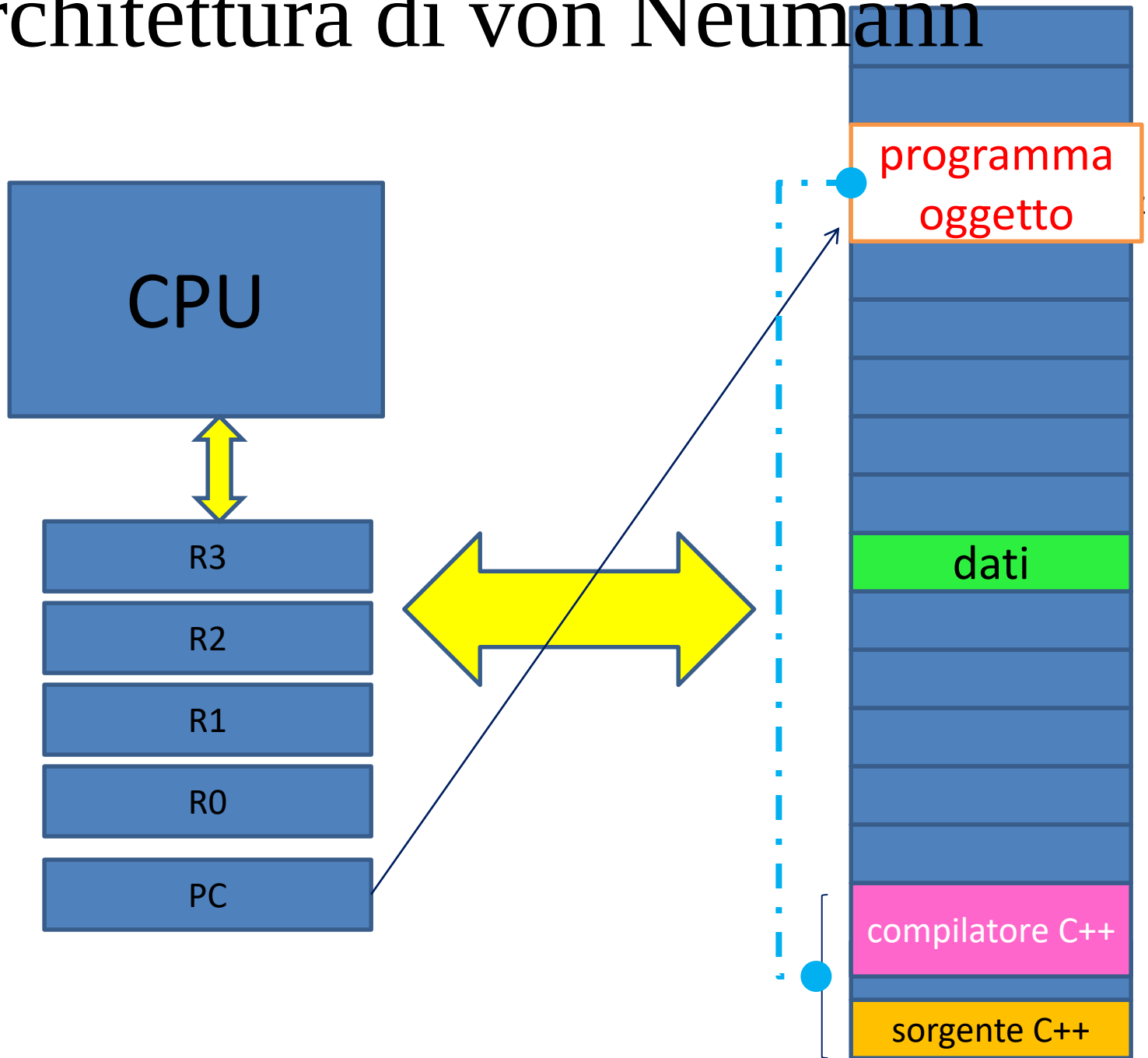
lunedì 8:30-10:30, lum 250
martedì 8:30-10:30, Lab 140 Paolotti
mercoledì 10:30-12:30, lum 250

- moodle per esercizi, compitini ed esami
- 3 compitini danno bonus fino a 1
- 4 esercizi a tempo con bonus ½ punto
- si passa con scritto almeno 18 + bonus

Cosa abbiamo fatto nella I parte

- architettura di von Neumann
- Linguaggio C++ minimale (cap. 2 e 3 del testo)
- qualche programma con dimostrazioni di correttezza (cap. 4 del testo)
- uso del laboratorio per gli esercizi e gli esami

architettura di von Neumann



Il C(++) in una nocciolina

- tipi : char, int e bool 2.1
- dichiarazioni 2.2
- input/output >> e << Cap 3
- assegnazione
- condizionale
- while

Capitolo 2: Tipi predefiniti

- già usati : char, int e bool
- float e double
- void
- short int, long int

ogni tipo occupa un certo numero di byte della memoria

per sapere quanti:

int s= sizeof(tipo)

- char occupa 1 byte = 8 bit

256 caratteri

contiene i caratteri ASCII = 128 caratteri

+ altri 128

- anche bool occupano 1 byte

- int occupano 4 byte

- float (4 byte)

- double (8 byte)

usano codifica floating point:

[+/-, mantissa, esponente]
rappresenta valore :

+/- 1.mantissa * (2 ^ esponente)

attenzione

rappresentazione degli interi (complemento a 2) e dei reali (floating point)
sono completamente diverse

ma alla fine ogni valore è sempre costituito da alcuni byte con dei bit che hanno valorer 0 o 1

devo sapere il **tipo** del valore per interpretare la sequenza di bit nel modo giusto

ogni variabile deve venire dichiarata prima di essere usata:

```
int x; // indefinita attenzione
double y; // indefinita attenzione
float  z= 10; // con inizializzazione, ma ??
```

conversione automatica 10 -> 10.0f

ogni variabile ha R- ed L-valore

per esempio: float z=23.3f

R-valore di z è 23.3 float
L-valore è l'indirizzo RAM in cui si trova questo valore

```
z = z * 2;
L-      R-valore di z
```

&z è un’espressione che ha come valore l’L-valore di z

espressione

12 - x * pippo + y

si vuole sempre calcolare il valore di un'espressione, cioè eseguire le operazioni

domande:

- c'è una costante intera 12 e tre variabili, devono essere intere?
- in che ordine si eseguono le operazioni?
- quale +, - e * si eseguono?

l’ordine dipende dalla precedenza degli operatori e dall’associatività:

12 - pippo * x + y
(12 - (pippo*x)) + y

* ha precedenza su + e - che associano a sinistra

per le altre domande vari casi:
-se x, y e pippo sono tutti int allora si usano le operazioni +, - e * tra int

-e se pippo fosse char, x float e y double ?

(12 - (pippo*x)) + y

char -> float pippo in float
int -> float 12 in float
float -> double risultato è double

conversioni automatiche

principio di base delle conversioni automatiche:

si converte valore di tipo che usa meno byte in valore «equivalente» di tipo che usa più byte

non si perdono informazioni (o quasi)

promozioni

char, bool -> int -> float -> double

assumiamo che pippo sia char, x float e y double

(12 - (pippo*x)) + y

- * tra float
- - tra float
- + tra double

overloading degli operatori:
operazioni diverse sono rappresentate dallo stesso simbolo, *, +, -, ecc.
overloading = sovraccaricamento

a + b

il tipo di a e b viene sempre reso uguale con conversioni

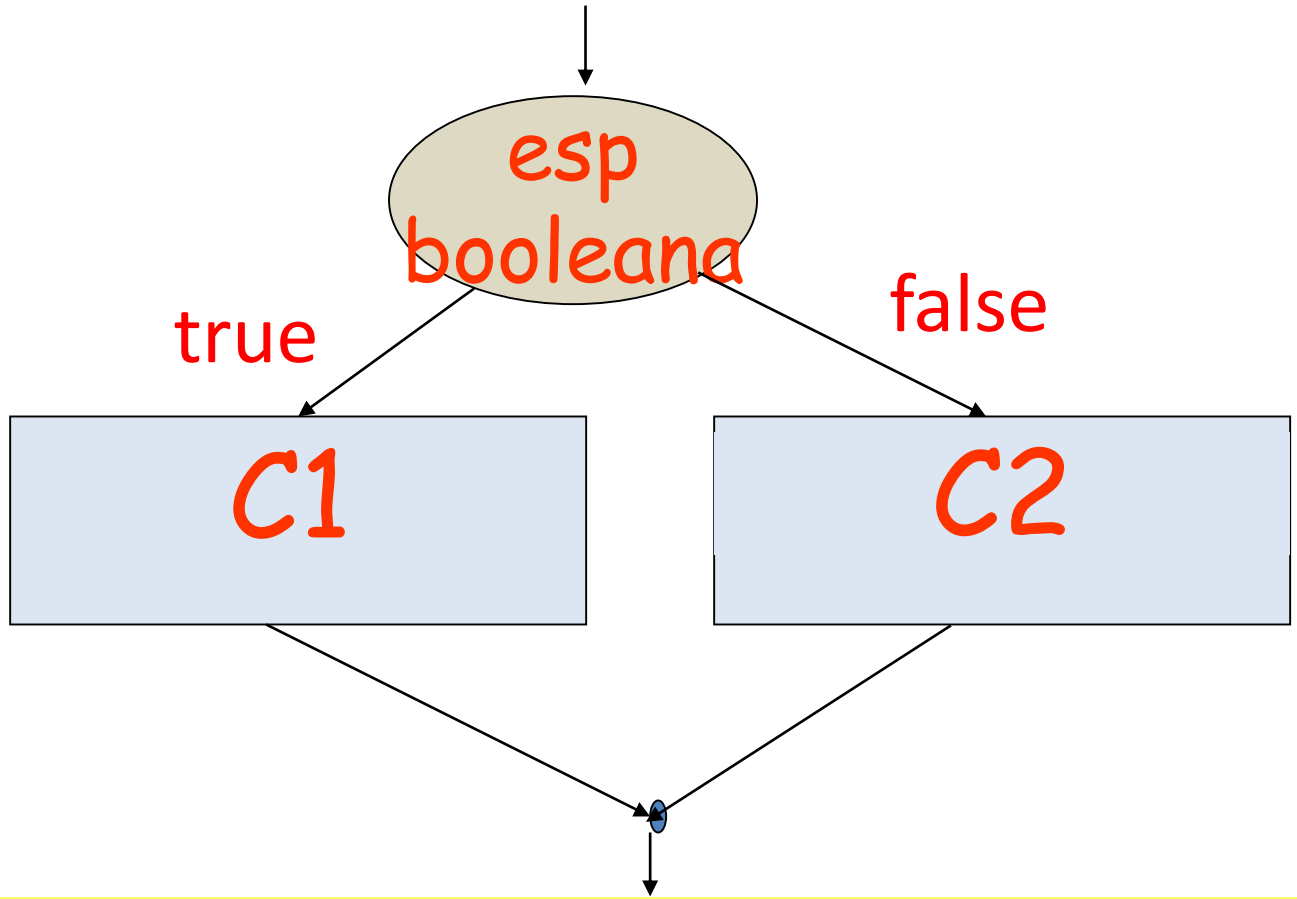
e poi si usa + giusta per quel tipo

test su
letsfeedback.com
login = wayca

istruzioni di base del C++

- I/O >> e <<
- assegnazione, x = espressione;
- condizionale
- ciclo while

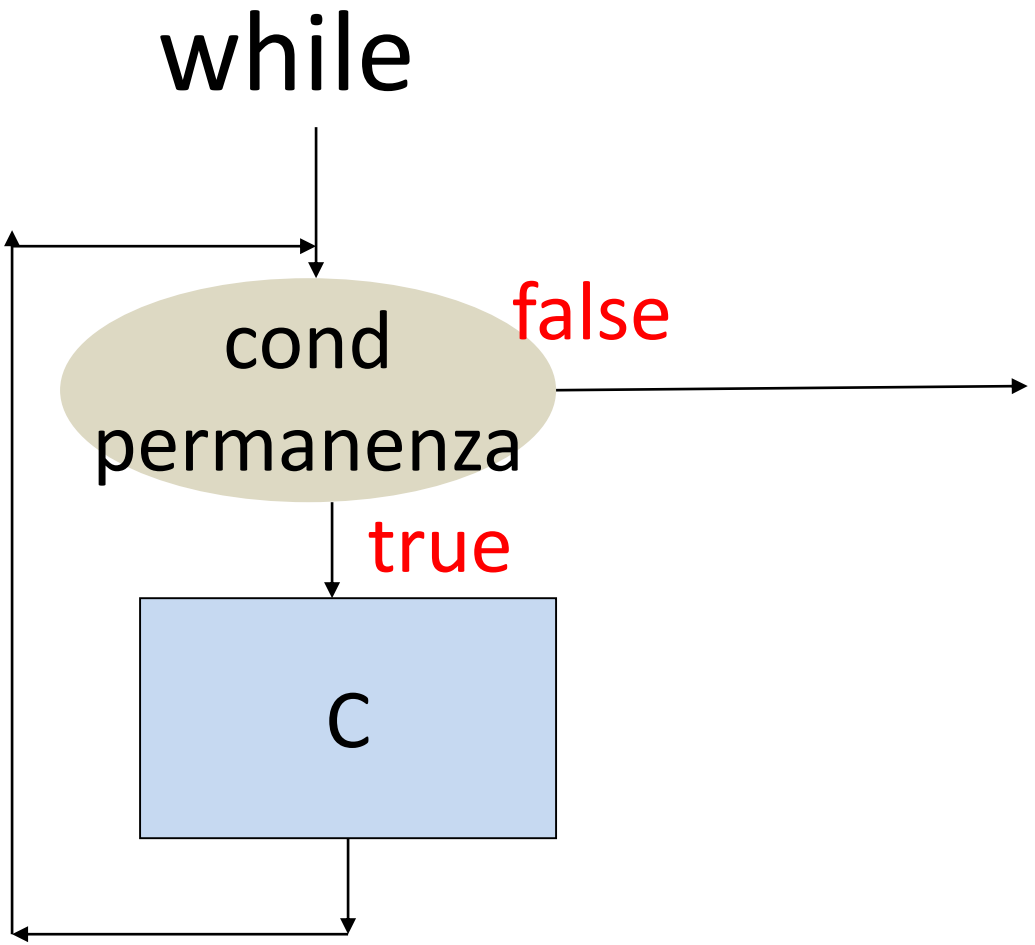
condizionale



ramo then

ramo else

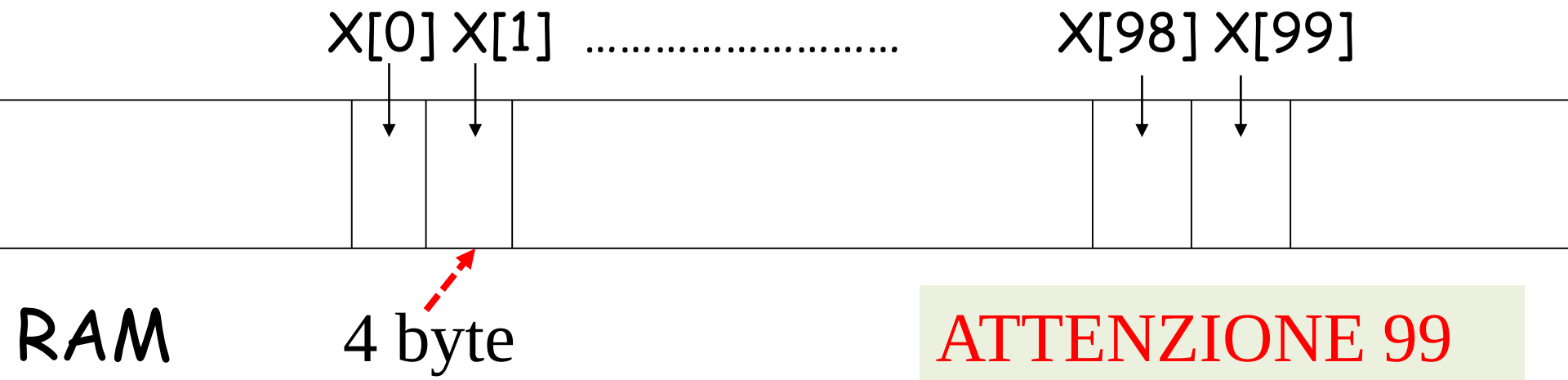
1 punto d'entrata ed 1 d'uscita



1 punto d'entrata ed 1 d'uscita

array

```
int X[100];
```



ATTENZIONE 99

lettura da cin di un array

PRE=(cin contiene n>0 seguito da n interi)

```
main()
{
    int n, A[100],i=0;
    cin >> n;
    if(n>100)
       cout<<″Troppi valori ″ <<endl;
    else
      while(i<n)
         {cin >> A[i]; i=i+1;}
}
```

lettura stampa di un array:

```
int n, A[100],i=0;
cin >> n;
while(i<n)
   {cin >> A[i]; i=i+1;}
i=0;
while(i<n)
     {cout<<A[i]<<‘ ‘; i = i+1;}
```

```
//ricerca del massimo
int n, A[100], i=0, max;
cin >> n;
while(i<n)
   {cin >> A[i]; i=i+1;}
i=1; posmax=0;
while(i<n)
  {
    if(A[i] > A[posmax])
          posmax=i;
    i=i+1;
    } //POST=(A[posmax] è il massimo in A[0..n-1])
```