

Iterazioni

(for, do, while)

Iterazioni

istruzioni che consentono di eseguire un *loop* (ciclo):

while

do...while

for

con alcune differenze non solo sintattiche...

Sintassi di while

while (espressione) **istruzione**

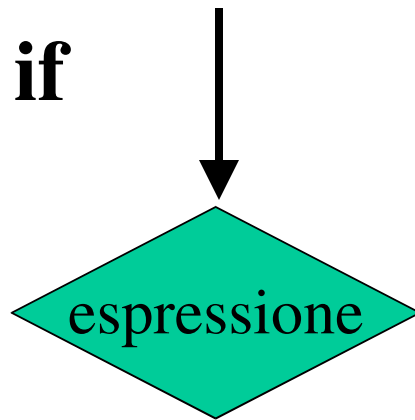
dove **espressione** è una qualsiasi espressione C e **istruzione** può essere una singola istruzione o una sequenza di istruzioni racchiusa tra { e }.

Semantica di `while`

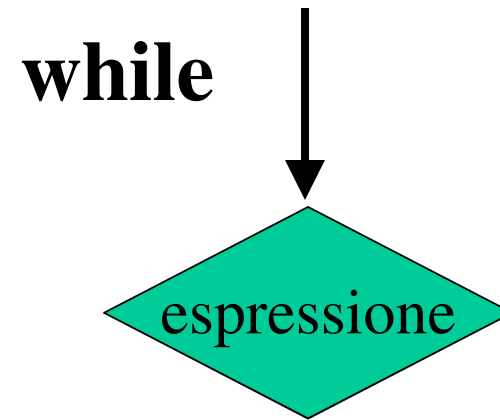
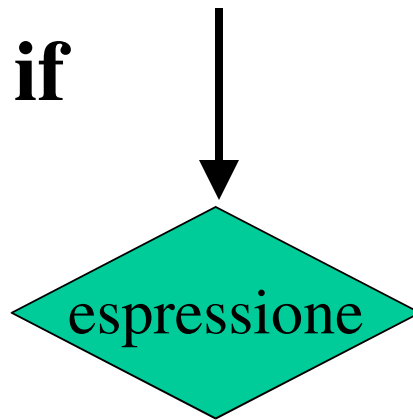
Nell'esecuzione di un'istruzione `while` viene

1. Valutata l'espressione `espressione`
 1. Se non è nulla si esegue l'`istruzione`
 2. Se è nulla si passa alle istruzioni successive al ciclo `while`
2. Si torna al punto 1

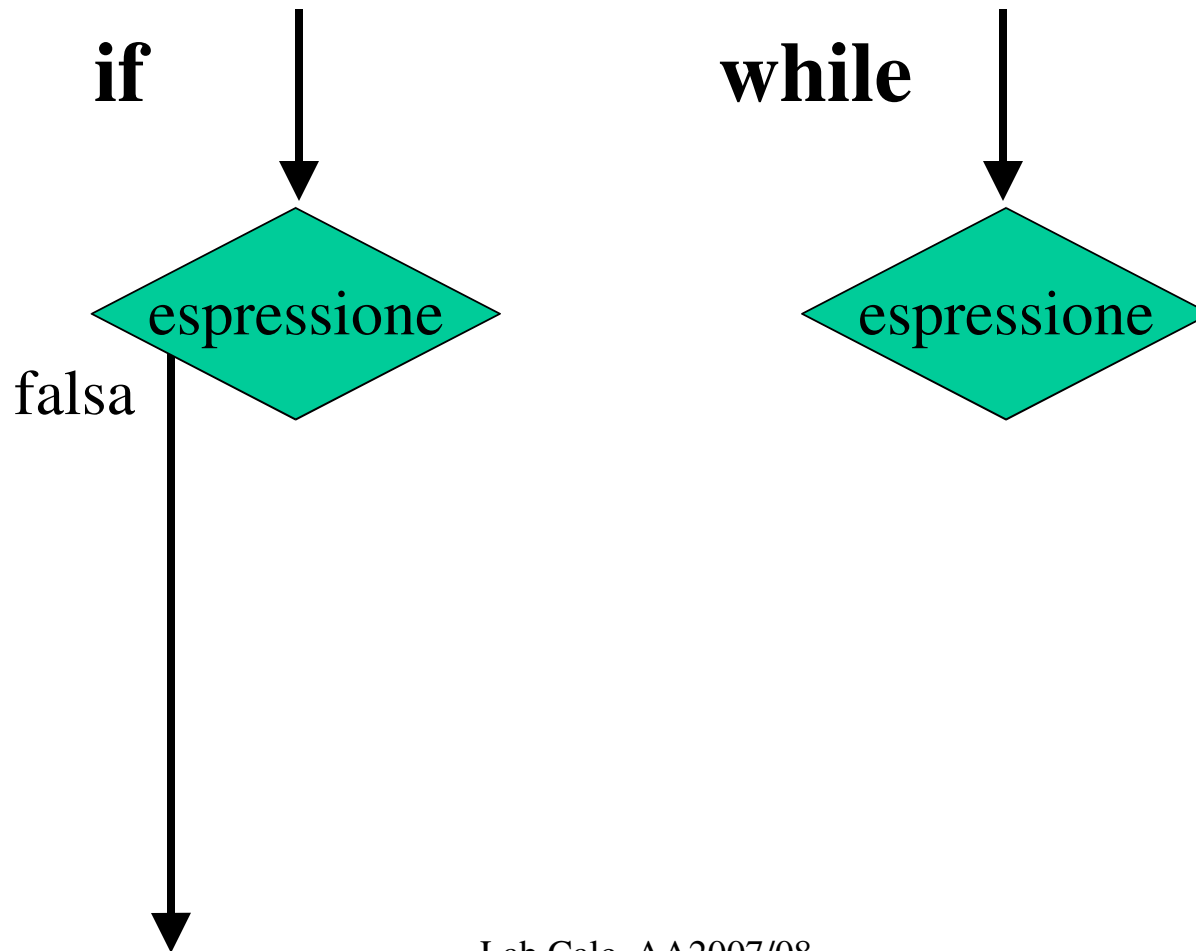
Confronto tra if e while



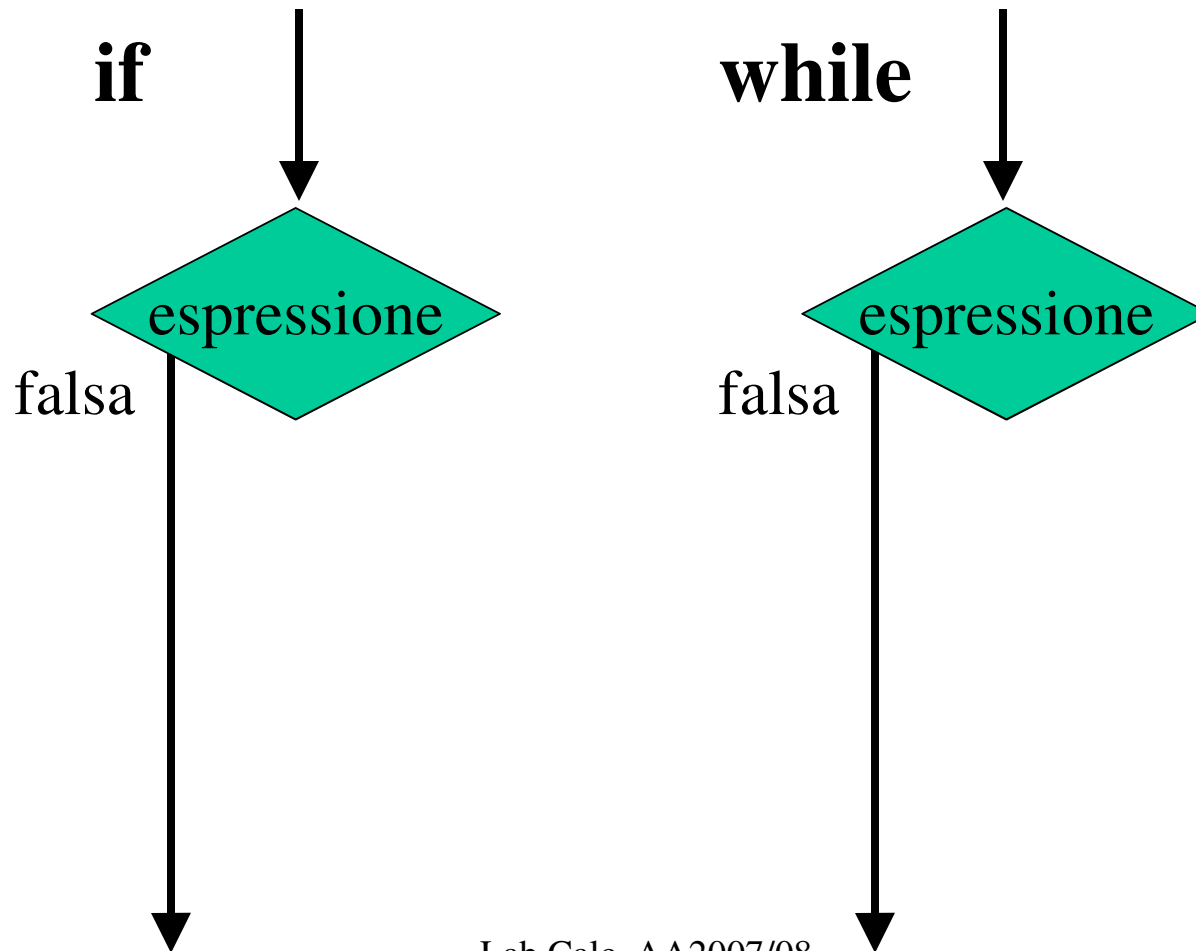
Confronto tra if e while



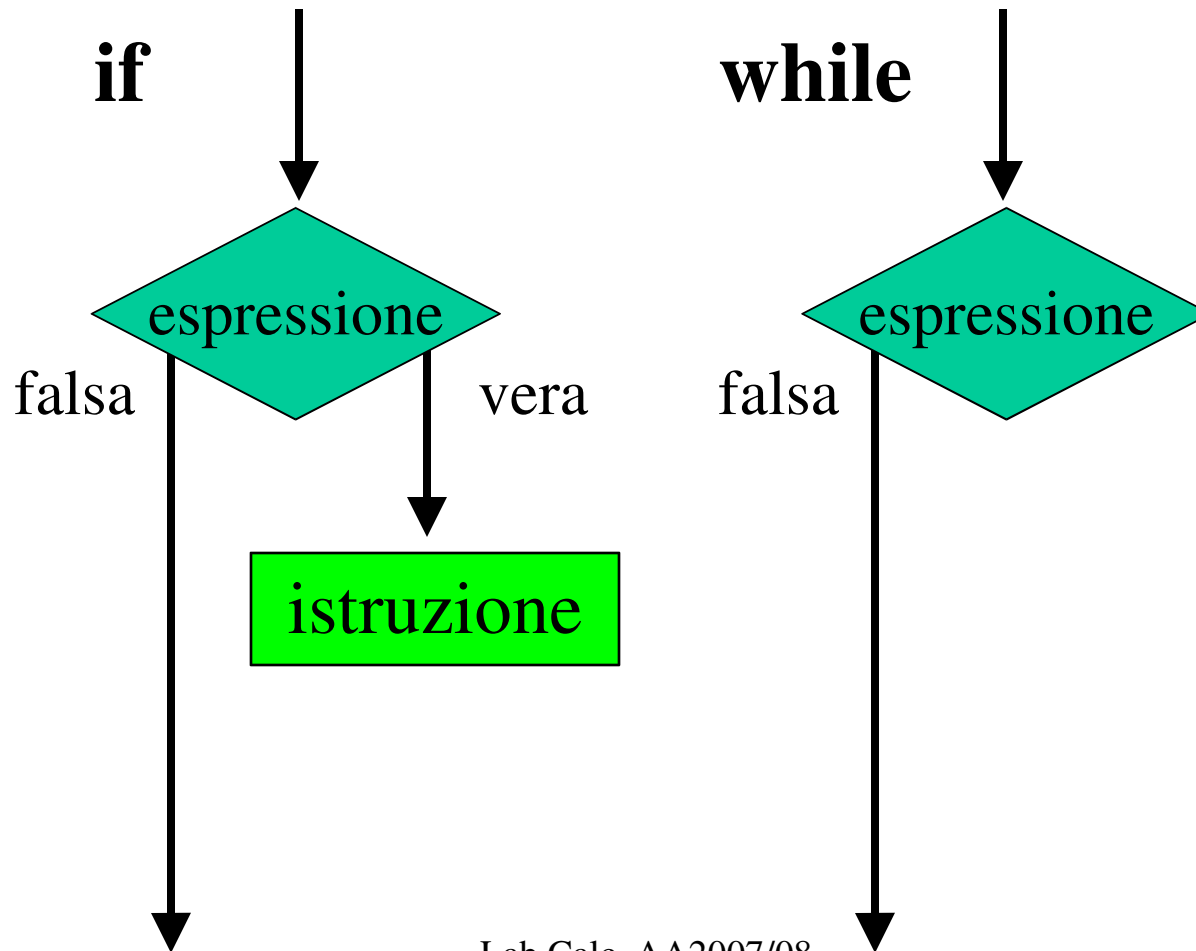
Confronto tra if e while



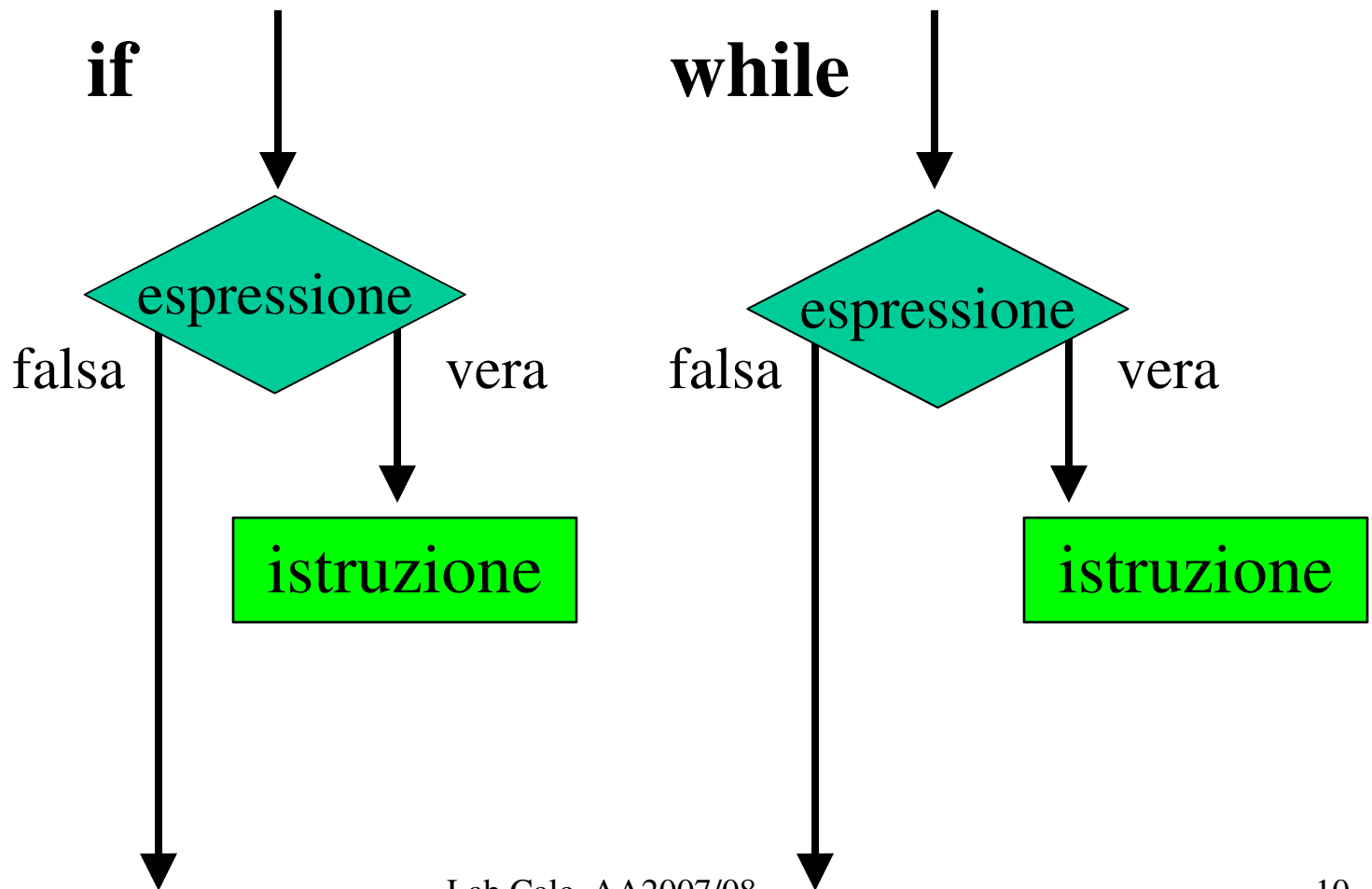
Confronto tra if e while



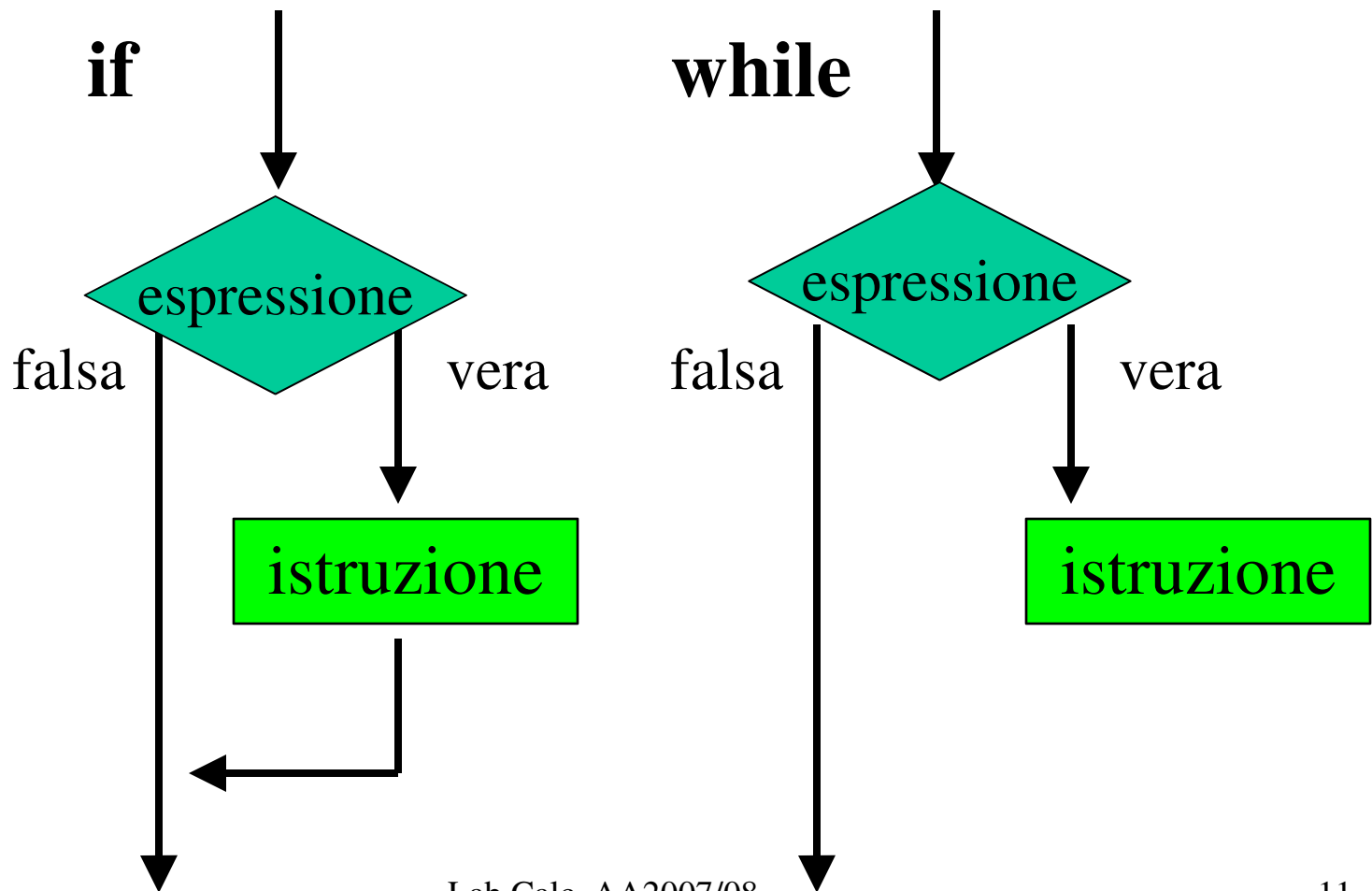
Confronto tra if e while



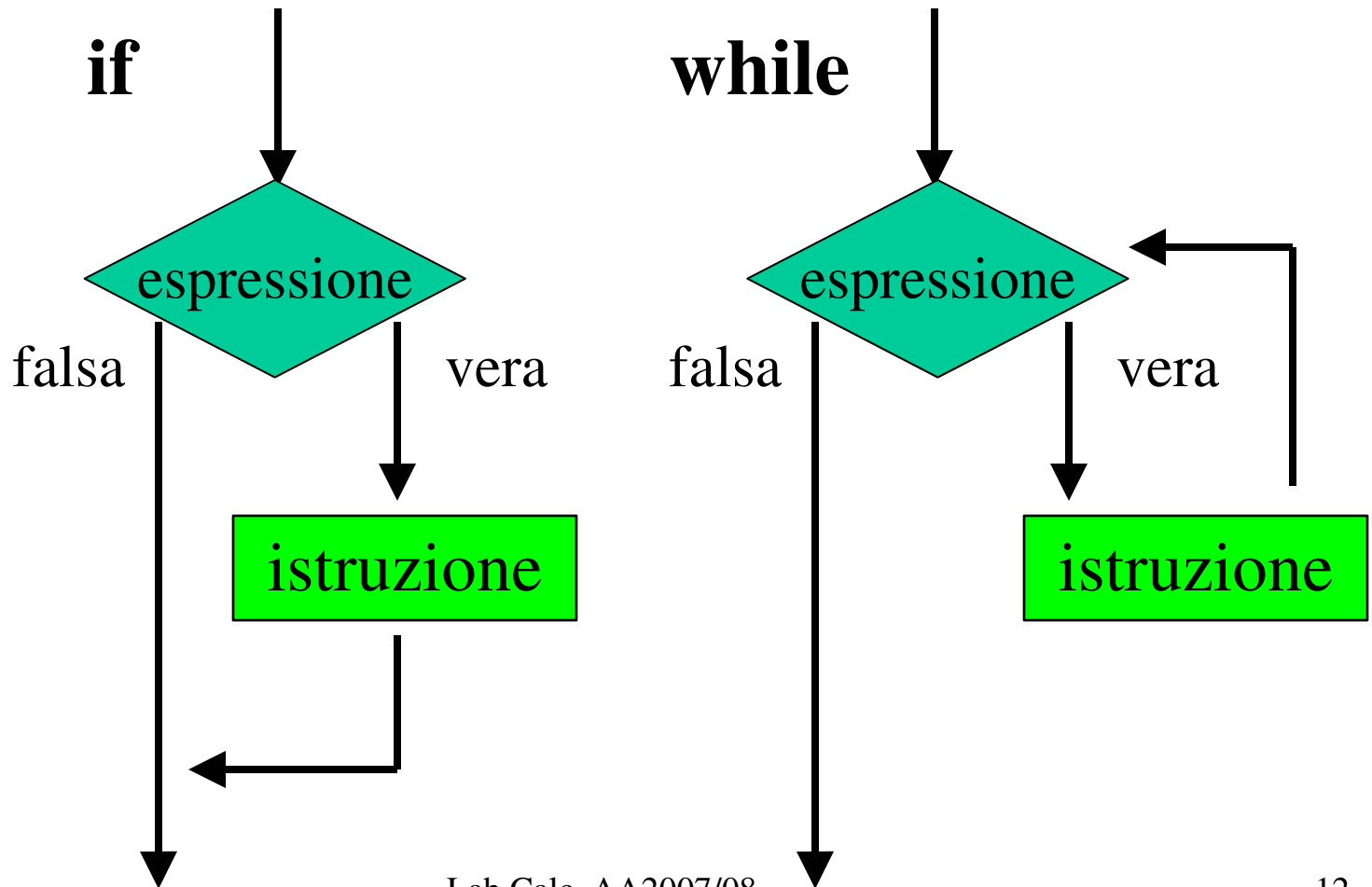
Confronto tra if e while



Confronto tra if e while



Confronto tra if e while



Confronto tra if e while

while è un if insistente!

Esempio 1 di uso di while

```
int contatore = 0;
int somma = 0;
while(contatore <= 25) {
    somma = somma + contatore;
    contatore = contatore + 1;
}
```

Esempio 2 di uso di while

```
double somma= 0;
int contatore =0;
while(1){      /* sempre vero! */
    if (contatore > 25) break; /*esce dal ciclo*/
    somma = somma + contatore;
    contatore++;
}
```

/* NOTA 1

come già detto l'uso di break andrebbe evitato
quindi l'esempio precedente è migliore

NOTA 2

per uscire da un'iterazione di un ciclo senza
uscire dal ciclo stesso si può utilizzare
l'istruzione `continue` */

Sintassi di do...while

do **istruzione** while (**espressione**)

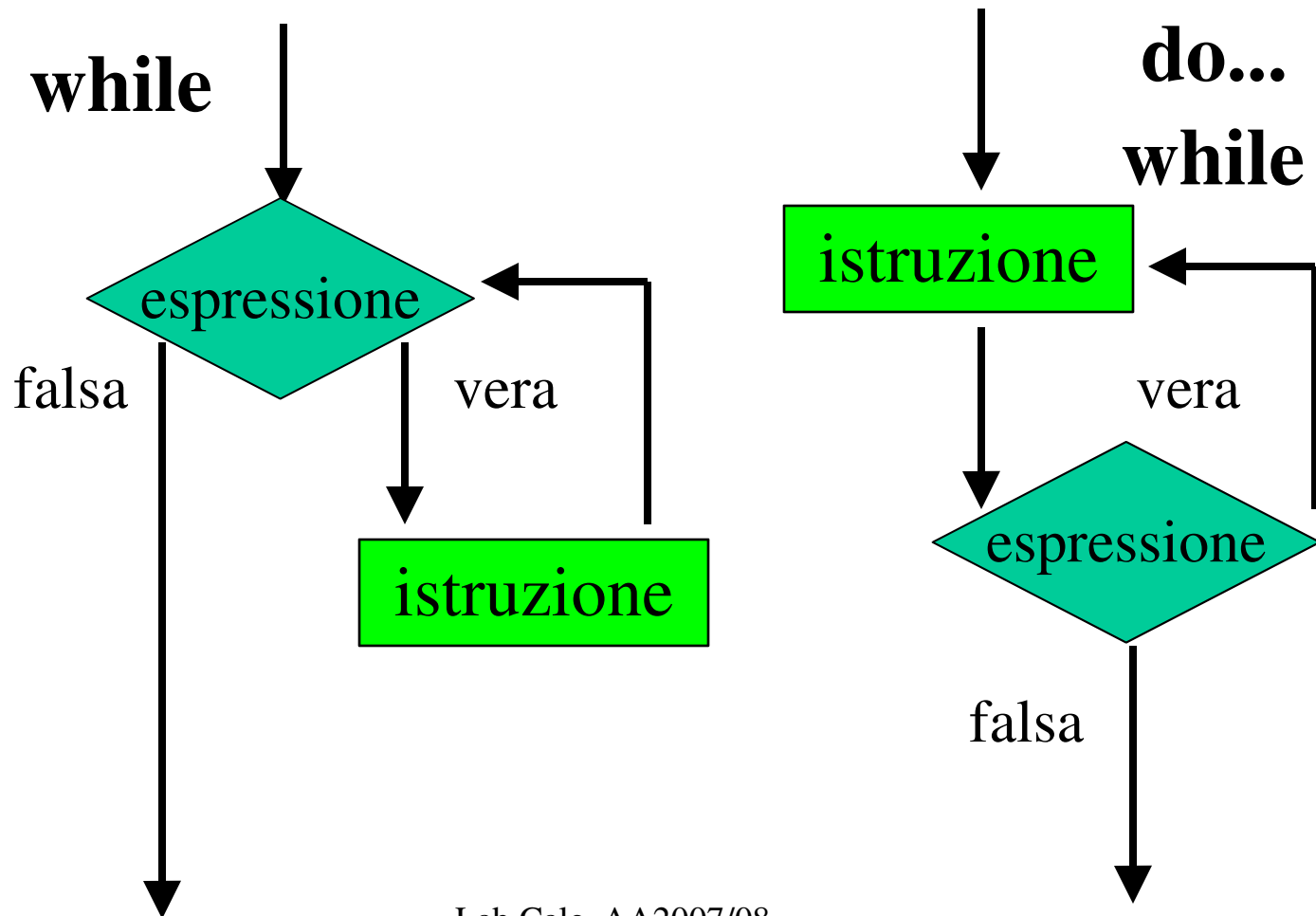
dove **espressione** è una qualsiasi espressione C e **istruzione** può essere una singola istruzione o una sequenza di istruzioni racchiusa tra { e }.

Semantica di do...while

Nell'esecuzione di un'istruzione do...while viene

1. Eseguita l'**istruzione**
2. Valutata l'**espressione** tra parentesi:
 1. Se non è nulla si torna al punto 1
 2. Se è nulla si passa alle istruzioni successive al do...while

Confronto tra while e do...while



Confronto tra while e do...while

do...while esegue sempre l'istruzione
almeno una volta

Esempio di uso di do...while

```
do {  
    printf("Inserisci la tua eta' \n");  
    scanf("%d",&age);  
    if(age <= 0){  
        printf("Deve essere un numero positivo!\n") ;  
    }  
} while(age <= 0);
```

Sintassi di for

```
for (espr1;espr2;espr3) istruzione
```

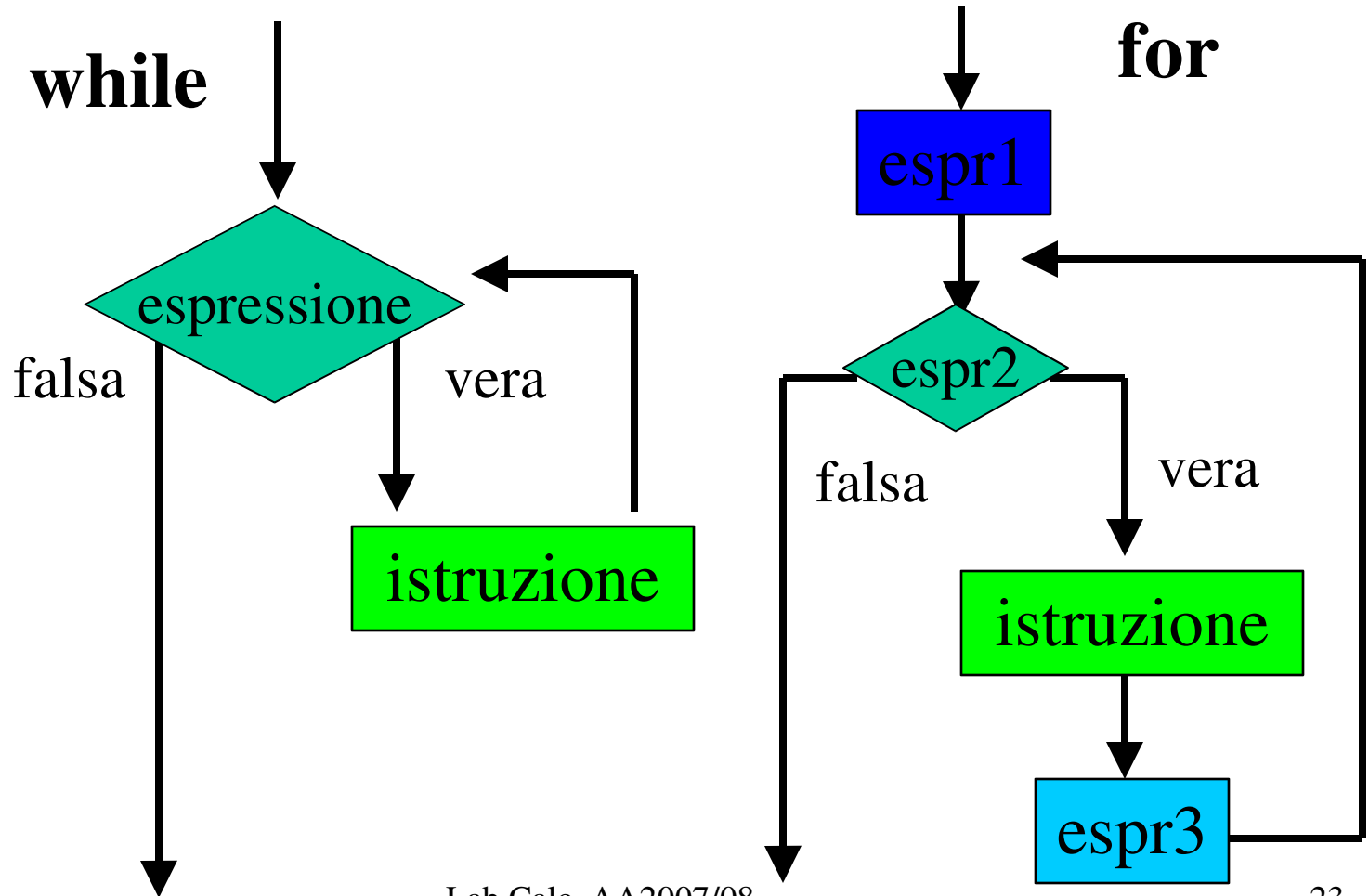
dove `espr1` `espr2` e `espr3` sono espressioni C e `istruzione` può essere una singola istruzione o una sequenza di istruzioni racchiusa tra `{ e }`.

Semantica di for

Nell'esecuzione di un'istruzione **for** viene

1. Valutata l'espressione **espr1** (di solito **inizializzazione del contatore: esempio $i=0$**)
2. Valutata l'espressione **espr2** (di solito un test del contatore: **esempio $i<10$**)
 - Se non è nulla si esegue l'**istruzione**
 - Se è nulla si passa alle istruzioni successive al ciclo **for**
3. Valutata l'espressione **espr3** (di solito un incremento o decremento del contatore, **esempio $i++$**)
4. Si torna al punto 2

Confronto tra while e for



Confronto tra while e for

```
for (espr1;espr2;espr3) istruzione
```

Si può riscrivere anche come

```
espr1;  
while (espr2) {  
    istruzione;  
    espr3;  
}
```


Esempio 1 di uso di for

```
int sum=0;
int i;
int n=100;
for (i = 1; i <= n; i=i +1) {
    sum = sum + i;
}
printf("La somma dei primi %d interi vale %d\n",n,sum);
```

Esempio 2 di uso di for: calcolo del fattoriale

```
int fattoriale=1;
int i, n;
printf("inserisci n\n");
scanf("%d",&n);
for(i=1; i<=n; i++) {
    fattoriale *= i;
}
printf("%d!=%d",n,fattoriale);
/* fattoriale, intero a 32 bit, puo' contenere solo
   valori fino a 12! per arrivare a 20! bisogna
   cambiare rappresentazione ed usare per esempio degli
   unsigned long long int (%llu). Non standard! */
```