IL COSTRUTTO SEQUENZA

La programmazione strutturata

Le istruzioni sono scritte una di seguito all'altra nell'ordine in cui vogliamo che siano interpretate dal compilatore e poi eseguite.

Non sono contemplate istruzioni di salto (Basic).

Pascal, C

Strutture
- sequenza
- selezione
- iterazione

Esempio

Creare un programma che, ricevuto un numero intero in input ne calcola la metà e visualizza il risultato.

```
#include <iostream>
using namespace std;
main(){
int n; float m;
cout<<‘‘Inserisci un numero intero  ’’;
cin >> n;
m=n/2.0;  //oppure m=(float)n/2;
cout<<‘‘\nLa metà del numero inserito è ’’<< m;}
```

Testing e debugging

Il testing di un programma è l'insieme di tutte le operazioni che vengono eseguite dopo aver finito di scrivere il codice al fine di:

- Verificare che il programma sia adeguato alle specifiche richieste
- Effettuare prove di funzionamento ed escludere la presenza di eventuali errori.

Testing e debugging

Il debugging è l'insieme delle tecniche che consentono di individuare ed eliminare gli errori.

Verificare la sintassi delle istruzioni

Verificare la sequenza di istruzioni

Mettere opportune istruzioni di stampa

- per vedere quale valore assumono le variabili
- per vedere se arriva ad eseguire certe istruzioni

Trace table

E’ una tabella che consente di annotare, su carta, le variazioni che ogni istruzione genera sul valore di una variabile.

Consente di individuare eventuali inopportune operazioni di sovrascrittura di una variabile o operazioni diverse da quelle richieste dall’algoritmo.

E’ importante simulare a mano l’esecuzione del codice per avere idea di cosa avverrà durante l’esecuzione.

Tipologie di errore

- Errori sintattici: errori sulla scrittura delle parole appartenenti al vocabolario del linguaggio usato. Rilevati dal compilatore o dall'interprete che ne segnala anche la posizione.
- Errori semantici: errori sulla sequenza o sulla scelta delle istruzioni. Il programma sarà sintatticamente corretto ma produrrà un risultato differente da quello atteso.
  - Errori run-time: dipendono dai dati inseriti, rilevabili in fase di esecuzione (ex. divisione per 0, uso di dati non compatibili, overflow, etc.).

Le variabili di lavoro

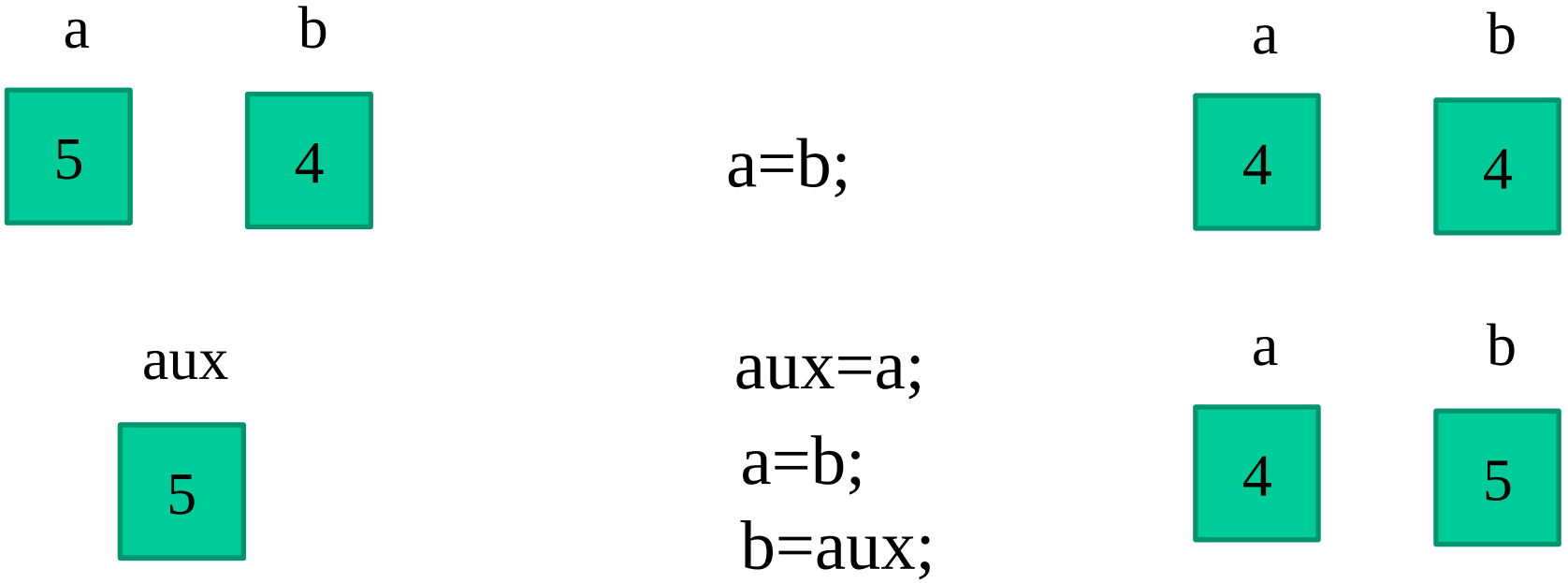
Sono variabili che vengono dichiarate ed usate solo per il corretto funzionamento dell'algoritmo e non per scambiare dati con l'esterno.

- Variabili ausiliarie
- Accumulatori
- Contatori

Le variabili ausiliarie

Servono per conservare temporaneamente il valore di una variabile destinata ad essere sovrascritta.

Scambiare il valore di due variabili.



Le variabili ausiliarie

Scambiare il valore di due variabili.

```
#include <iostream>
using namespace std;
main(){ int a=5; int b=4; int aux;
cout <<‘‘a=’’<< a << ‘‘e b=’’<< b;
aux=a;
a=b;
b=aux;
cout<<‘‘\nDopo lo scambio si ha a=’’<<a<< ‘‘e b=’’<< b;}
```

Le variabili ausiliarie

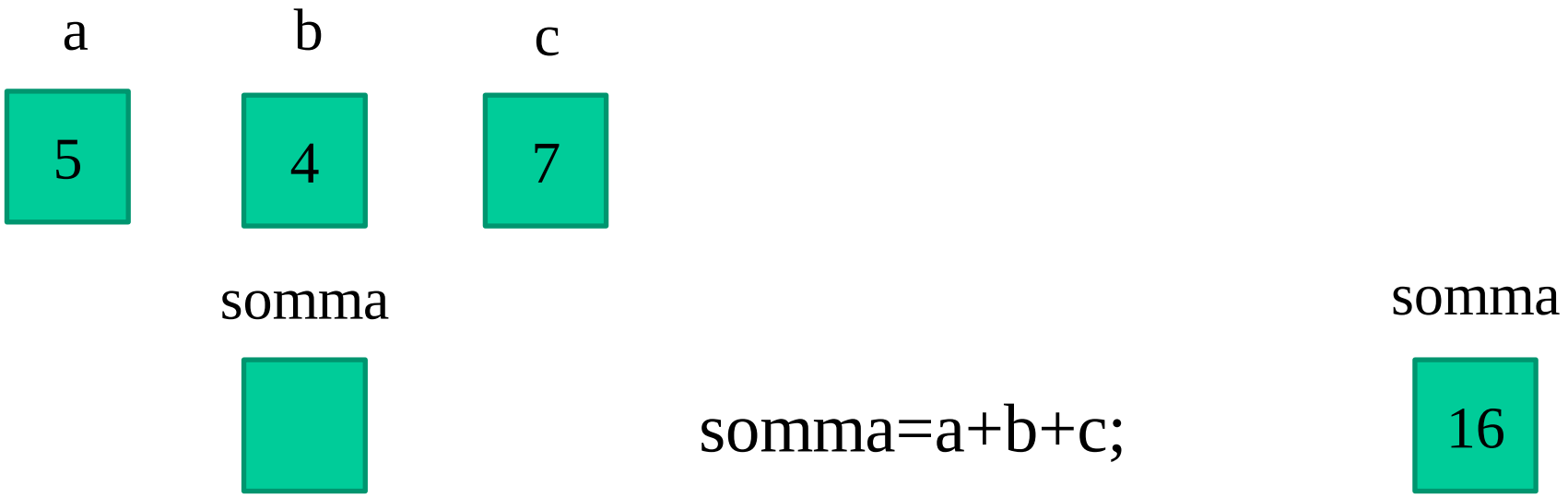
Scambiare il valore di due variabili.

```
#include <iostream>
main(){ int a=5; int b=4; int aux;
cout <<‘‘I valori delle due variabili sono a=’’ << a <<  ‘‘e b=’’ <<b;
aux=a;
a=b;
b=aux;
cout <<‘‘\nDopo lo scambio si ha a=’’ << a <<  ‘‘e b=’’ <<b;
```

Gli accumulatori

Servono per conservare somme (o più in generale risultati parziali di operazioni aritmetiche) di valori acquisiti da tastiera o generati durante l'elaborazione.

Sommare 3 numeri interi acquisiti da tastiera (senza accumulatore).



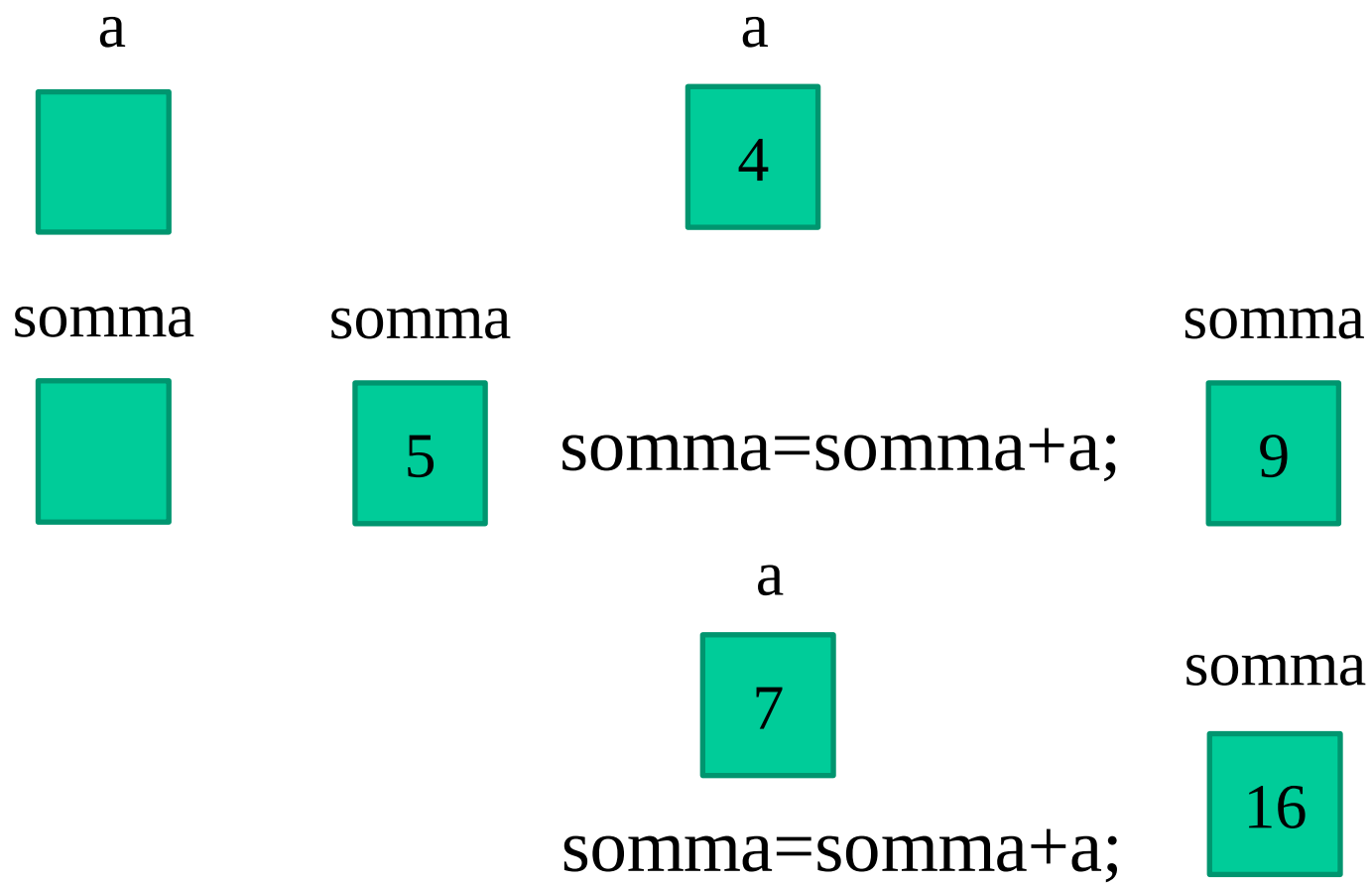
Gli accumulatori

Sommare 3 numeri interi acquisiti da tastiera

```
#include <iostream>
using namespace std;
main(){ int a,b,c; int somma;
cout << ‘‘Inserire 3 numeri interi’’;
cin >> a >> b>> c;
somma=a+b+c;
cout <<‘‘\nLa somma dei 3 numeri è ’’<<somma;
}
```

Gli accumulatori

Sommare 3 numeri interi acquisiti da tastiera (con l'accumulatore).



Gli accumulatori

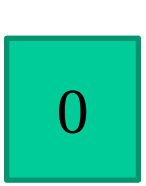
Sommare 3 numeri interi acquisiti da tastiera

```
#include <iostream>
using namespace std;
main(){ int a; int somma;
cout <<‘‘Inserire un numero intero’’;
cin >> somma;
cout <<‘‘\nInserire un numero intero’’; cin >>a;
somma=somma+a;
cout <<‘‘\nInserire un numero intero’’; cin >>a;
somma=somma+a;
cout <<‘‘\nLa somma dei 3 numeri è ’’<<somma;}
```

I contatori

Servono per contare il susseguirsi di una serie di operazioni che si ripetono. Per usare un contatore sarà necessario:

1. Dichiarare una variabile ed **inizializzare il suo contenuto** al valore da cui si vuole iniziare a contare (in genere 0).



2. Aggiornare il contenuto della variabile sommando le unità necessarie (in genere 1)

```
conta=conta+1;          conta++;
```

I contatori

Stampare 3 volte il proprio nome e contare le operazioni eseguite. Visualizzare il risultato.

```
#include <iostream>
using namespace std;
main(){int conta=0;
cout <<‘‘Maria Cristina’’;
conta++;
cout <<‘‘ Maria Cristina’’; conta++;
cout <<‘‘ Maria Cristina’’; conta++;
cout <<‘‘\nHo eseguito’’<< conta <<‘‘operazioni’’;}
```