

Linguaggi di Programmazione I – Lezione 2

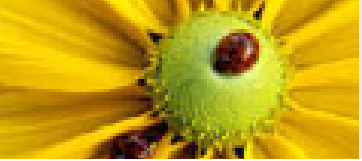
Prof. Marco Faella

<mailto://m.faella@unina.it>

<http://wpage.unina.it/mfaella>

Materiale didattico elaborato con i Proff. Sette e Bonatti

20 marzo 2023



Panoramica della lezione

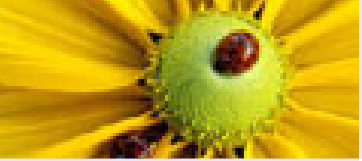
Modello imperativo

Data Object e legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione



Modello imperativo

Memoria
Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Modello imperativo



Memoria

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Consiste in un insieme di “contenitori di dati” ...
 - ◆ Ad es. parole (o celle) della memoria centrale
 - ◆ Tipicamente rappresentate dal loro indirizzo



Memoria

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Consiste in un insieme di “contenitori di dati” ...
 - ◆ Ad es. parole (o celle) della memoria centrale
 - ◆ Tipicamente rappresentate dal loro indirizzo
- ... associati ai valori in essi contenuti
 - ◆ I valori delle variabili



Memoria

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Consiste in un insieme di “contenitori di dati” ...
 - ◆ Ad es. parole (o celle) della memoria centrale
 - ◆ Tipicamente rappresentate dal loro indirizzo
- ... associati ai valori in essi contenuti
 - ◆ I valori delle variabili
- Dunque (concettualmente) la memoria è
 - ◆ Una funzione da uno spazio di locazioni ad uno spazio di valori
 - ◆ $\text{mem} : \text{Indirizzi} \rightarrow \text{Valori}$
 - ◆ **$\text{mem}(\text{loc})$** = “valore contenuto nella locazione loc”



Grammatica in forma Backus-Naur

Modello imperativo

Memoria

**Grammatica in
forma Backus-Naur**

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Una sintassi per le grammatiche context-free
- Invece di “ $\text{exp} \rightarrow \text{exp} + \text{exp}$ ” ...
- ...scriveremo “ $\langle \text{exp} \rangle ::= \langle \text{exp} \rangle + \langle \text{exp} \rangle$ ”



Grammatica in forma Backus-Naur

[Modello imperativo](#)

[Memoria](#)

[Grammatica in
forma Backus-Naur](#)

[Assegnazioni](#)

[Modello imperativo](#)

[Ambiente](#)

[Esempi di ambiente:
imperativo](#)

[Esempi di ambiente:
funzionale](#)

[Esempio:
assegnazione](#)

[Esercizio](#)

[Data Object e
legami](#)

[Legami di tipo](#)

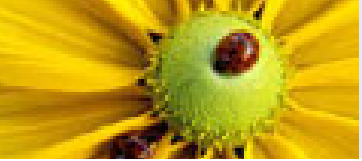
[Blocchi di istruzioni](#)

[Legami di locazione](#)

- Una sintassi per le grammatiche context-free
- Invece di “ $\text{exp} \rightarrow \text{exp} + \text{exp}$ ” ...
- ...scriveremo “ $\langle \text{exp} \rangle ::= \langle \text{exp} \rangle + \langle \text{exp} \rangle$ ”

Si ricordi che:

- “exp” è un *simbolo non-terminale*
- “+” è un *simbolo terminale*
- “ $\langle \text{exp} \rangle ::= \langle \text{exp} \rangle + \langle \text{exp} \rangle$ ” è una *produzione o regola*

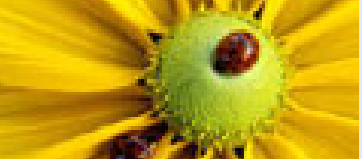


Assegnazioni

■ Definizione grammaticale (sintassi):

```
<assegnazione> ::= <name> <assignment-operator> <expression>
```

<name> rappresenta la locazione dove viene posto il risultato mentre in <expression> sono specificati una computazione e i riferimenti ai valori necessari alla computazione.



Assegnazioni

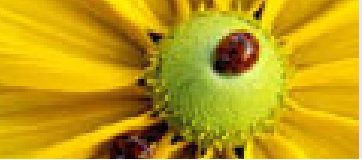
■ Definizione grammaticale (sintassi):

```
<assegnazione> ::= <name> <assignment-operator> <expression>
```

<name> rappresenta la locazione dove viene posto il risultato mentre in <expression> sono specificati una computazione e i riferimenti ai valori necessari alla computazione.

Esempio in Pascal:

```
a := b + c;
```



Assegnazioni

■ Definizione grammaticale (sintassi):

```
<assegnazione> ::= <name> <assignment-operator> <expression>
```

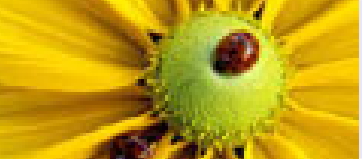
<name> rappresenta la locazione dove viene posto il risultato mentre in <expression> sono specificati una computazione e i riferimenti ai valori necessari alla computazione.

Esempio in Pascal:

```
a := b + c;
```

■ Esecuzione (significato, semantica):

Il valore di <expression> va memorizzato nell'indirizzo rappresentato da <name>.



Assegnazioni

■ Definizione grammaticale (sintassi):

```
<assegnazione> ::= <name> <assignment-operator> <expression>
```

<name> rappresenta la locazione dove viene posto il risultato mentre in <expression> sono specificati una computazione e i riferimenti ai valori necessari alla computazione.

Esempio in Pascal:

```
a := b + c;
```

■ Esecuzione (significato, semantica):

Il valore di <expression> va memorizzato nell'indirizzo rappresentato da <name>.

Il valore di <expression> dipenderà dai valori contenuti negli indirizzi degli argomenti di <expression> rappresentati dai nomi di questi ottenuto seguendo le prescrizioni del codice associato al suo nome.



Modello imperativo

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo
Esempi di ambiente:
funzionale
Esempio:
assegnazione
Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- È il più vicino agli elaboratori “standard”.



Modello imperativo

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo
Esempi di ambiente:
funzionale
Esempio:
assegnazione
Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- È il più vicino agli elaboratori “standard”.
- I programmi sono descrizioni di sequenze di modifiche della “memoria” del calcolatore.
 - ◆ può anche essere il modo di pilotare display ed altre periferiche: *memory-mapped IO*



Modello imperativo

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo
Esempi di ambiente:
funzionale
Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legame di locazione

- È il più vicino agli elaboratori “standard”.
- I programmi sono descrizioni di sequenze di modifiche della “memoria” del calcolatore.
 - ◆ può anche essere il modo di pilotare display ed altre periferiche: *memory-mapped IO*
- Ogni unità di esecuzione consiste di 4 passi:
 1. ottenere indirizzi delle locazioni di operandi e risultato;



Modello imperativo

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo
Esempi di ambiente:
funzionale
Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- È il più vicino agli elaboratori “standard”.
- I programmi sono descrizioni di sequenze di modifiche della “memoria” del calcolatore.
 - ◆ può anche essere il modo di pilotare display ed altre periferiche: *memory-mapped IO*
- Ogni unità di esecuzione consiste di 4 passi:
 1. ottenere indirizzi delle locazioni di operandi e risultato;
 2. ottenere dati di operandi da locazioni di operandi;



Modello imperativo

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo
Esempi di ambiente:
funzionale
Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- È il più vicino agli elaboratori “standard”.
- I programmi sono descrizioni di sequenze di modifiche della “memoria” del calcolatore.
 - ◆ può anche essere il modo di pilotare display ed altre periferiche: *memory-mapped IO*
- Ogni unità di esecuzione consiste di 4 passi:
 1. ottenere indirizzi delle locazioni di operandi e risultato;
 2. ottenere dati di operandi da locazioni di operandi;
 3. valutare risultato;



Modello imperativo

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo
Esempi di ambiente:
funzionale
Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- È il più vicino agli elaboratori “standard”.
- I programmi sono descrizioni di sequenze di modifiche della “memoria” del calcolatore.
 - ◆ può anche essere il modo di pilotare display ed altre periferiche: *memory-mapped IO*
- Ogni unità di esecuzione consiste di 4 passi:
 1. ottenere indirizzi delle locazioni di operandi e risultato;
 2. ottenere dati di operandi da locazioni di operandi;
 3. valutare risultato;
 4. memorizzare risultato in locazione risultato.

In sostanza un assegnamento!



Modello imperativo

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo
Esempi di ambiente:
funzionale
Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- È il più vicino agli elaboratori “standard”.
- I programmi sono descrizioni di sequenze di modifiche della “memoria” del calcolatore.

◆ può anche essere il modo di pilotare display ed altre periferiche: *memory-mapped IO*

- Ogni unità di esecuzione consiste di 4 passi:
 1. ottenere indirizzi delle locazioni di operandi e risultato;
 2. ottenere dati di operandi da locazioni di operandi;
 3. valutare risultato;
 4. memorizzare risultato in locazione risultato.

In sostanza un assegnamento!

- Il modello imperativo usa dei nomi come astrazione di indirizzi di locazioni di memoria (variabili).



Ambiente (di esecuzione)

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

L'*ambiente* (environment) comprende:

- Un **insieme di nomi** (di variabili, parametri, funzioni, etc.)
- ... associati a qualcosa da cui si può risalire al **valore** della variabile o del parametro.



Ambiente (di esecuzione)

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

L'*ambiente* (environment) comprende:

- Un **insieme di nomi** (di variabili, parametri, funzioni, etc.)
- ... associati a qualcosa da cui si può risalire al **valore** della variabile o del parametro.
- Concettualmente l'ambiente è: una funzione da un insieme di identificatori (i nomi) a un insieme di ...?
 $\text{env}(\text{id}) = ???$



Ambiente (di esecuzione)

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

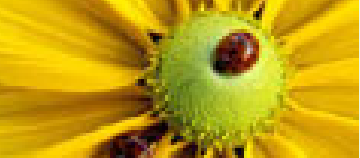
Legami di tipo

Blocchi di istruzioni

Legami di locazione

L'*ambiente* (environment) comprende:

- Un **insieme di nomi** (di variabili, parametri, funzioni, etc.)
- ... associati a qualcosa da cui si può risalire al **valore** della variabile o del parametro.
- Concettualmente l'ambiente è: una funzione da un insieme di identificatori (i nomi) a un insieme di ...?
 $\text{env}(\text{id}) = ???$
- Il codominio della funzione dipende dal paradigma computazionale del linguaggio.



Esempi di ambiente: imperativo

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

**Esempi di ambiente:
imperativo**

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Nel paradigma imperativo, la funzione env associa gli identificatori a locazioni di memoria, le quali, a loro volta, sono associate (funzione mem) al contenuto di memoria
- $\text{env: Nomi} \rightarrow \text{Indirizzi}$
- $\text{env}(x) = \text{indirizzo della variabile di nome } x$
- Il valore di una variabile x è $\text{mem}(\text{env}(x))$



Esempi di ambiente: imperativo

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

**Esempi di ambiente:
imperativo**

Esempi di ambiente:
funzionale

Esempio:
assegnazione

Esercizio

Data Object e
legami

Legami di tipo

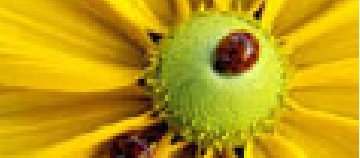
Blocchi di istruzioni

Legami di locazione

- Nel paradigma imperativo, la funzione `env` associa gli identificatori a locazioni di memoria, le quali, a loro volta, sono associate (funzione `mem`) al contenuto di memoria
- `env`: Nomi \rightarrow Indirizzi
- **`env(x)`** = indirizzo della variabile di nome `x`
- Il valore di una variabile `x` è `mem(env(x))`
- Attenzione: `env(x)` è immutabile finchè `x` esiste...

◆ vedremo che gli identificatori nascono e muoiono durante l'esecuzione

nel paradigma imperativo la funzione `env` identifica una associazione *immutabile* (la locazione di memoria associata a un nome non cambia);



Esempi di ambiente: funzionale

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

**Esempi di ambiente:
funzionale**

Esempio:

assegnazione

Esercizio

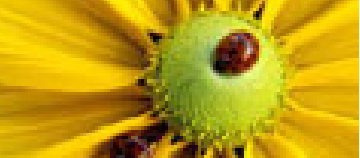
Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Nel paradigma funzionale (puro), il valore associato a un nome non cambia nel tempo
- Quindi, la funzione env associa direttamente gli identificatori al contenuto della memoria
- env: Nomi \rightarrow Valori
- Il valore di una variabile x è $\text{env}(x)$



Esempi di ambiente: funzionale

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo

**Esempi di ambiente:
funzionale**

Esempio:
assegnazione
Esercizio

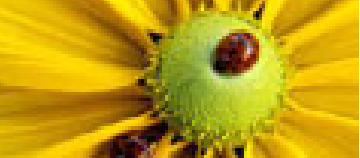
Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Nel paradigma funzionale (puro), il valore associato a un nome non cambia nel tempo
- Quindi, la funzione env associa direttamente gli identificatori al contenuto della memoria
- env: Nomi \rightarrow Valori
- Il valore di una variabile x è $\text{env}(x)$
- Anche in questo caso, env è una associazione immutabile (fintanto che la variabile x è in scope)



Esempi di ambiente: funzionale

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur
Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo

**Esempi di ambiente:
funzionale**

Esempio:
assegnazione

Esercizio

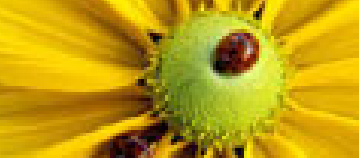
Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Nel paradigma funzionale (puro), il valore associato a un nome non cambia nel tempo
- Quindi, la funzione env associa direttamente gli identificatori al contenuto della memoria
- env: Nomi \rightarrow Valori
- Il valore di una variabile x è $\text{env}(x)$
- Anche in questo caso, env è una associazione immutabile (fintanto che la variabile x è in scope)
- Nel paradigma funzionale non esiste la funzione mem



Esempio: assegnazione

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

**Esempio:
assegnazione**

Esercizio

Data Object e
legami

Legami di tipo

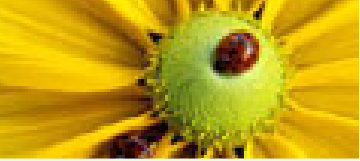
Blocchi di istruzioni

Legami di locazione

In una assegnazione:

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

il valore dell'espressione a destra va salvato nella variabile a sinistra
quindi:



Esempio: assegnazione

Modello imperativo

Memoria
Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente
Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

**Esempio:
assegnazione**

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

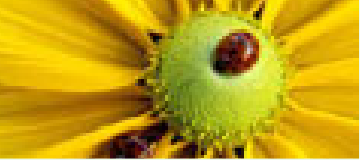
Legami di locazione

In una assegnazione:

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

il valore dell'espressione a destra va salvato nella variabile a sinistra
quindi:

la x di sinistra indica la locazione associata al nome (cioè $\text{env}(x)$)



Esempio: assegnazione

Modello imperativo

Memoria

Grammatica in
forma Backus-Naur

Assegnazioni

Modello imperativo

Ambiente

Esempi di ambiente:
imperativo

Esempi di ambiente:
funzionale

**Esempio:
assegnazione**

Esercizio

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

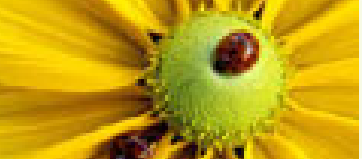
In una assegnazione:

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

il valore dell'espressione a destra va salvato nella variabile a sinistra
quindi:

la x di sinistra indica la locazione associata al nome (cioè $\text{env}(x)$)

la x di destra indica il valore della variabile (cioè $\text{mem}(\text{env}(x))$)



Esercizio

Date le seguenti definizioni di variabili nel linguaggio C:

```
int a[3] = { 4, 5, 6 };  
int *p, *q;
```

tradurre i seguenti assegnamenti in termini di `mem` ed `env`:

1. `a[2] = a[2] + 1`
2. `p = a`
3. `q = p`
4. `q = &a[1]`
5. `q = p + 2`

Viceversa, tradurre in C i seguenti assegnamenti:

1. In `mem(env(p))` store 0
2. In `env(a)+1` store `mem(env(a))`
3. In `mem(env(p))+1` store 0
4. In `mem(env(p))+mem(env(a))` store `mem(env(p))`



Modello imperativo

**Data Object e
legami**

Data Object: un
altro modo di
astrarre
l'implementazione
dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

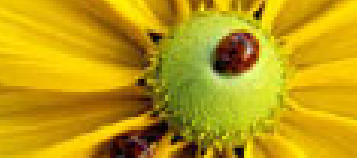
Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Data Object e legami



Data Object: un altro modo di astrarre l'implementazione dei dati

[Modello imperativo](#)

[Data Object e legami](#)

[Data Object: un altro modo di astrarre l'implementazione dei dati](#)

[Legami](#)

[Modifiche di legami](#)

[Esempio 1](#)

[Esempio 2](#)

[Esempio 3](#)

[Il puntatore \(1\)](#)

[Il puntatore \(2\)](#)

[Data object in C](#)

[Legami di tipo](#)

[Blocchi di istruzioni](#)

[Legami di locazione](#)

- Ogni variabile, parametro, oggetto, ... può essere rappresentato con un *data object*
- Un *data object* è una quadrupla (L, N, V, T) , ove:
 - ◆ L : locazione.
 - ◆ N : nome.
 - ◆ V : valore.
 - ◆ T : tipo.



Data Object: un altro modo di astrarre l'implementazione dei dati

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Ogni variabile, parametro, oggetto, ... può essere rappresentato con un *data object*
- Un *data object* è una quadrupla (L, N, V, T) , ove:
 - ◆ L : locazione.
 - ◆ N : nome.
 - ◆ V : valore.
 - ◆ T : tipo.
- Un *legame* è la determinazione di una delle componenti.
 - ◆ ad esempio legame di locazione, di nome, di valore, di tipo



Data Object: un altro modo di astrarre l'implementazione dei dati

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- Ogni variabile, parametro, oggetto, ... può essere rappresentato con un *data object*
- Un *data object* è una quadrupla (L, N, V, T) , ove:
 - ◆ L : locazione.
 - ◆ N : nome.
 - ◆ V : valore.
 - ◆ T : tipo.
- Un *legame* è la determinazione di una delle componenti.
 - ◆ ad esempio legame di locazione, di nome, di valore, di tipo
- Nel paradigma imperativo, dato un data object (L, N, V, T)
 - ◆ per il legame di locazione vale $L = env(N)$
 - ◆ per il legame di valore vale $V = mem(env(N))$



Legami

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

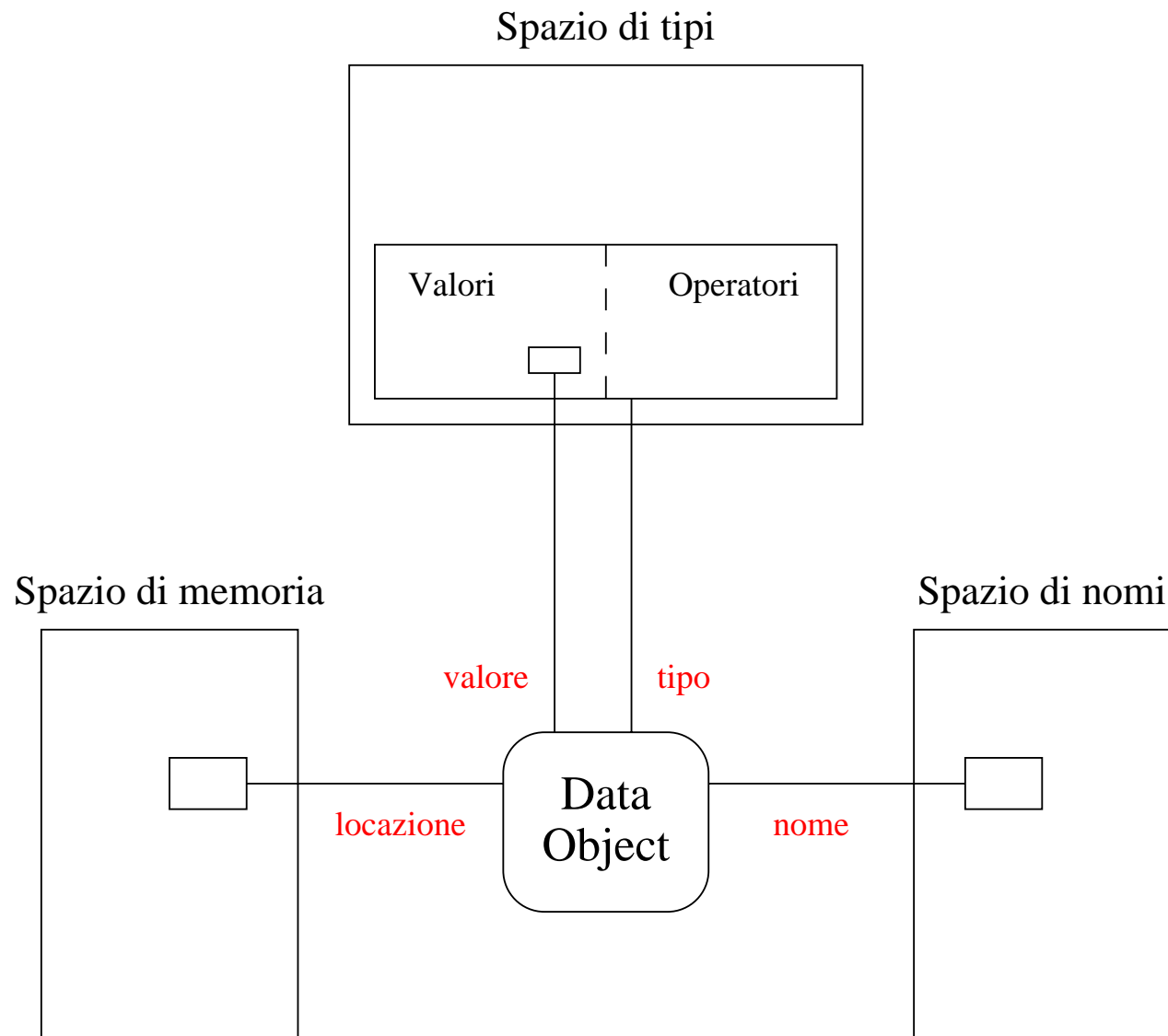
Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione





Modifiche di legami

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Variazioni di legami (binding) possono avvenire:

1. Durante la compilazione (compile time).



Modifiche di legami

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Variazioni di legami (binding) possono avvenire:

1. Durante la compilazione (compile time).
2. Durante il caricamento in memoria (load time).



Modifiche di legami

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Variazioni di legami (binding) possono avvenire:

1. Durante la compilazione (compile time).
2. Durante il caricamento in memoria (load time).
3. Durante l'esecuzione (run time).



Modifiche di legami

Modello imperativo

Data Object e
legami

Data Object: un
altro modo di
astrarre
l'implementazione
dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Variazioni di legami (binding) possono avvenire:

1. Durante la compilazione (compile time).
2. Durante il caricamento in memoria (load time).
3. Durante l'esecuzione (run time).

■ il *location binding* avviene durante il caricamento in memoria, oppure a run-time (si veda la gestione dei blocchi più avanti);



Modifiche di legami

Modello imperativo

Data Object e
legami

Data Object: un
altro modo di
astrarre
l'implementazione
dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Variazioni di legami (binding) possono avvenire:

1. Durante la compilazione (compile time).
2. Durante il caricamento in memoria (load time).
3. Durante l'esecuzione (run time).

- il *location binding* avviene durante il caricamento in memoria, oppure a run-time (si veda la gestione dei blocchi più avanti);
- il *name binding* avviene durante la compilazione, nell'istante in cui il compilatore incontra una dichiarazione;



Modifiche di legami

Modello imperativo

Data Object e
legami

Data Object: un
altro modo di
astrarre
l'implementazione
dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

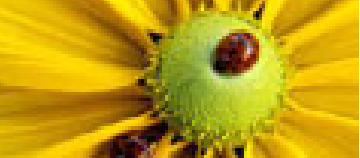
Blocchi di istruzioni

Legami di locazione

Variazioni di legami (binding) possono avvenire:

1. Durante la compilazione (compile time).
2. Durante il caricamento in memoria (load time).
3. Durante l'esecuzione (run time).

- il *location binding* avviene durante il caricamento in memoria, oppure a run-time (si veda la gestione dei blocchi più avanti);
- il *name binding* avviene durante la compilazione, nell'istante in cui il compilatore incontra una dichiarazione;
- il *type binding* avviene (di solito, si veda dopo) durante la compilazione, nell'istante in cui il compilatore incontra una dichiarazione di tipo; un tipo è definito dal sottospazio di valori (e dai relativi operatori) che un *data object* può assumere.



Esempio 1

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

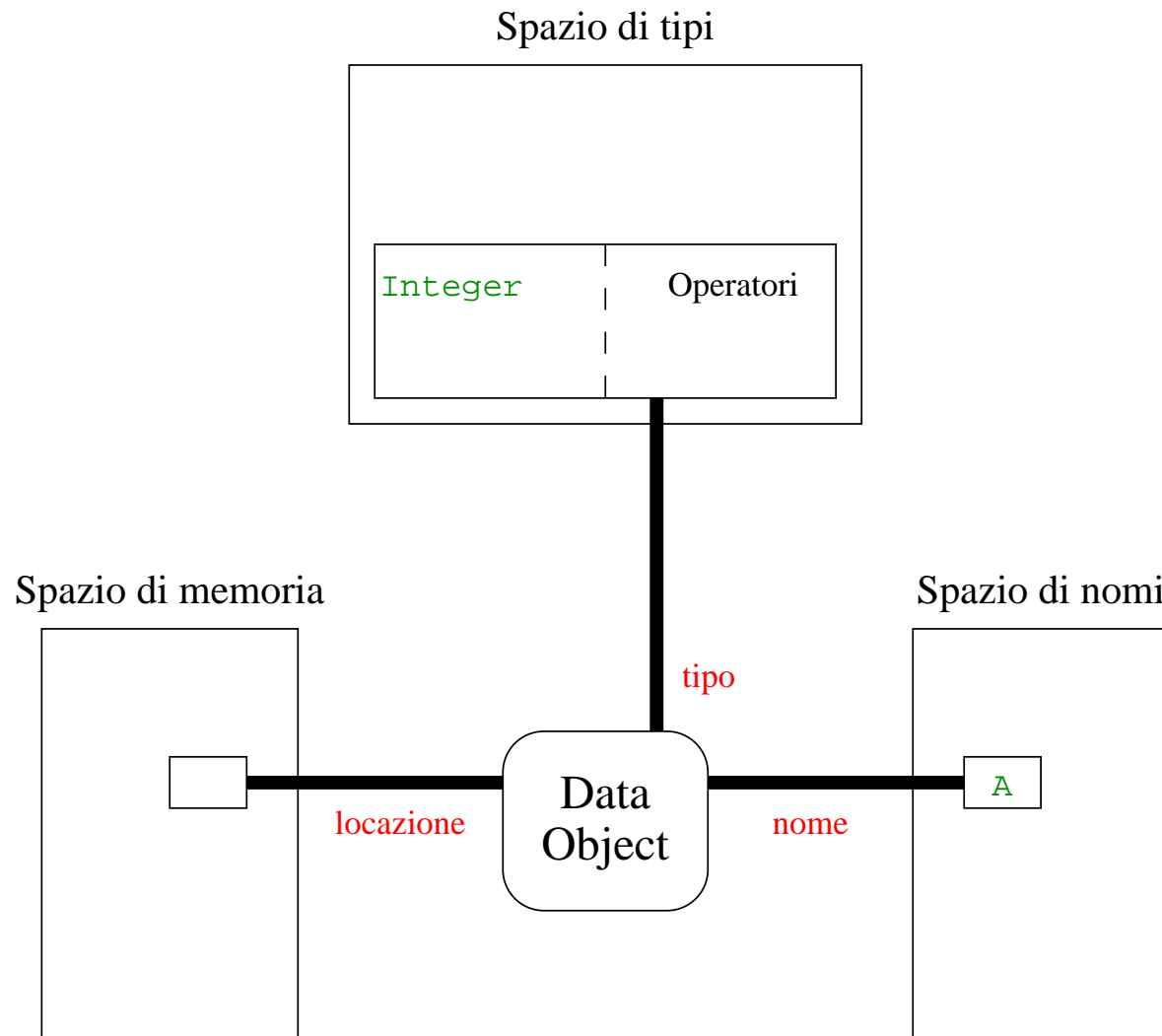
Il puntatore (2)

Data object in C

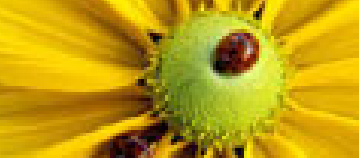
Legami di tipo

Blocchi di istruzioni

Legami di locazione



A: integer;



Esempio 2

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

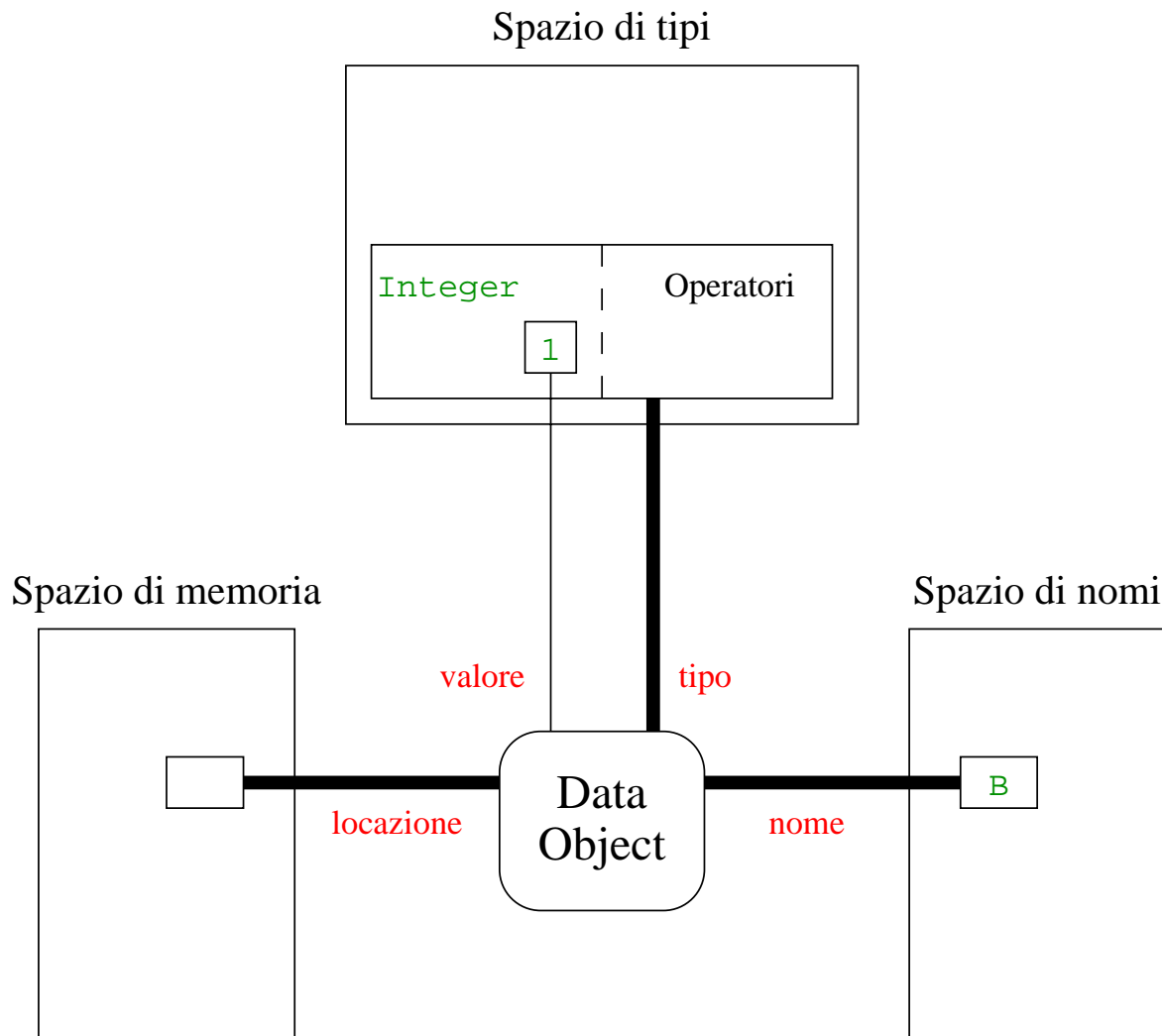
Il puntatore (2)

Data object in C

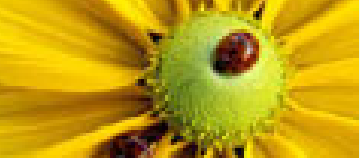
Legami di tipo

Blocchi di istruzioni

Legami di locazione



`B: integer:= 1;`



Esempio 3

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

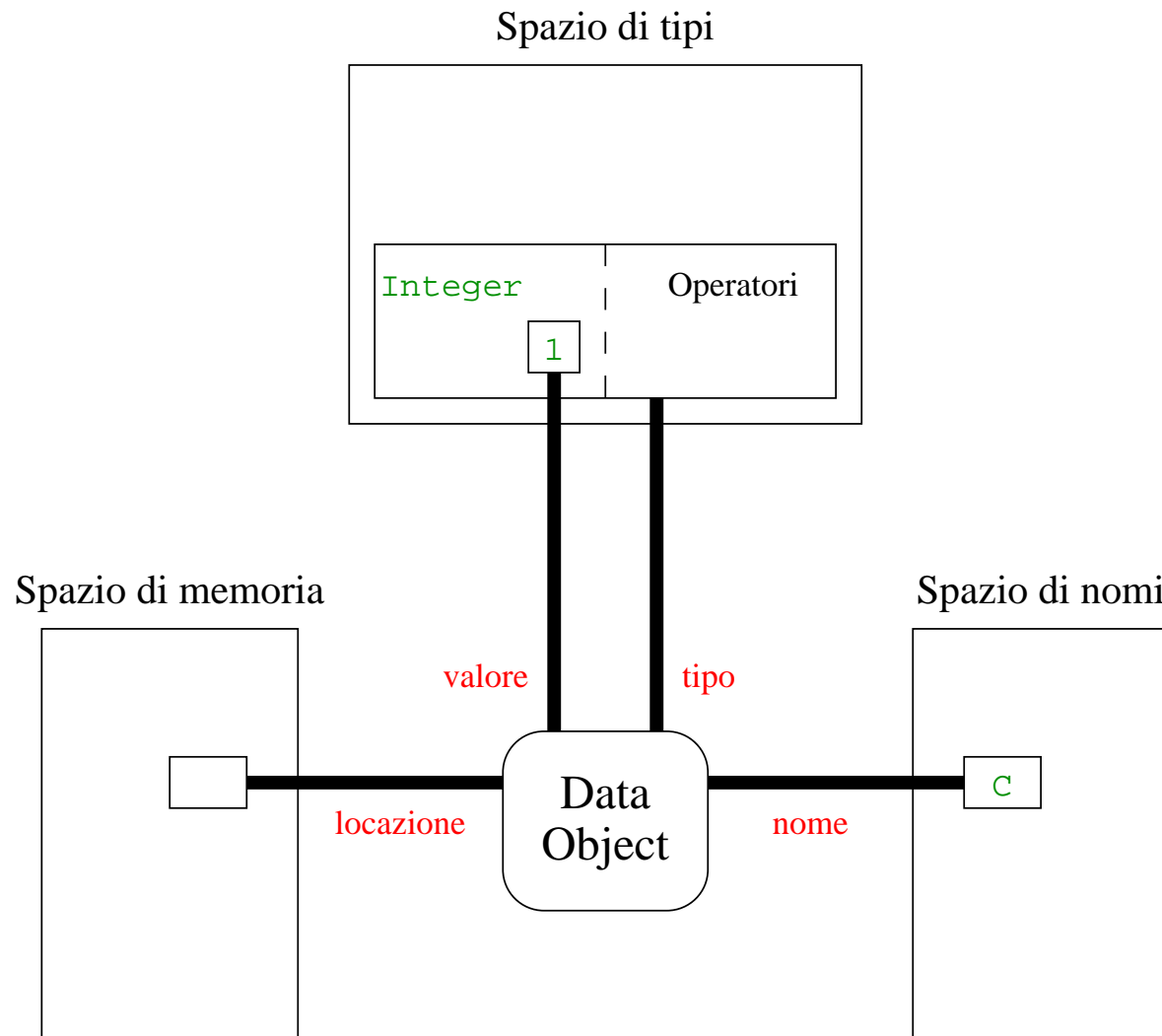
Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione



`C: constant integer:= 1;`



Il puntatore (1)

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

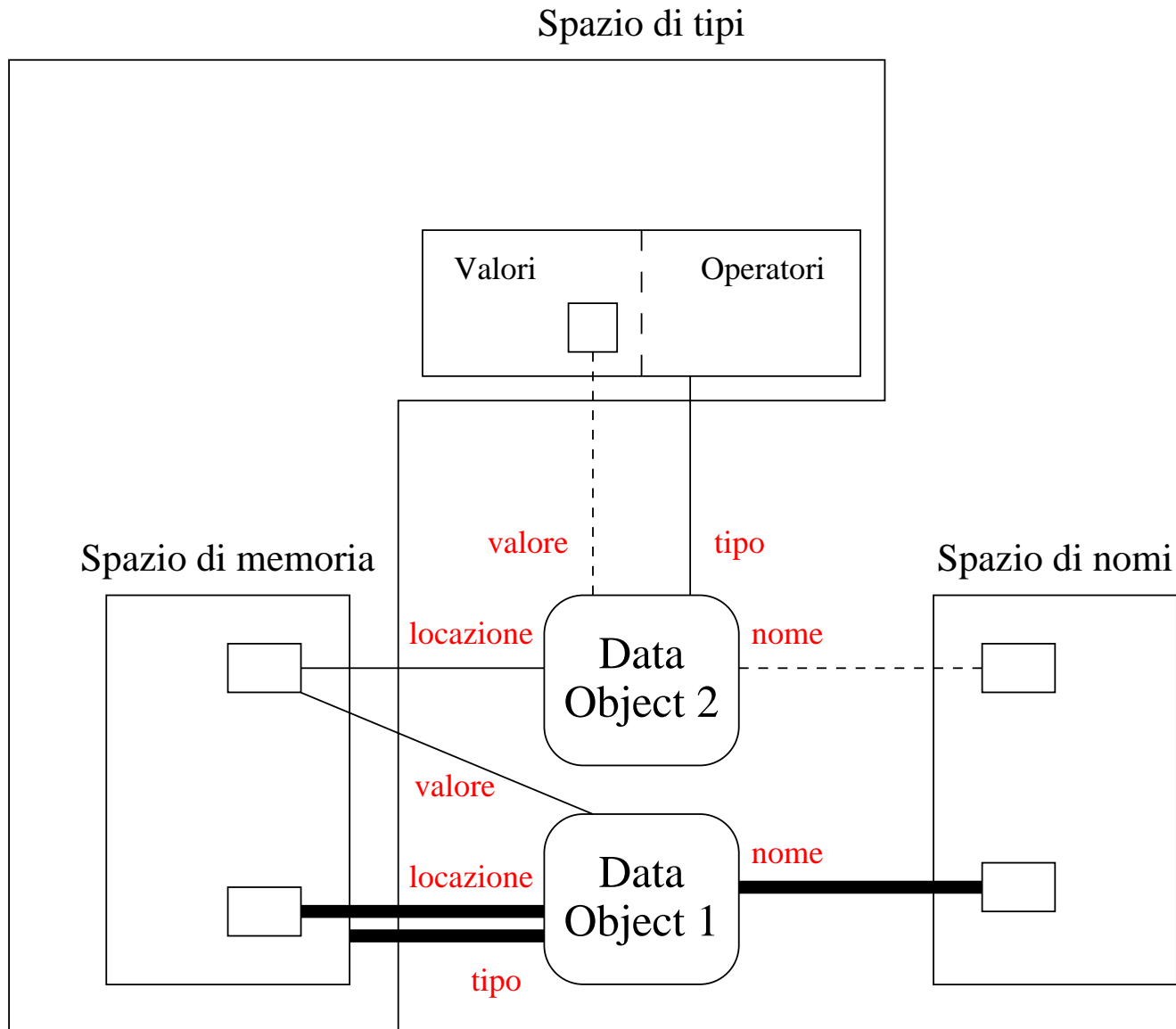
Il puntatore (2)

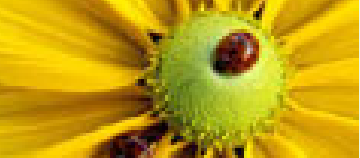
Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione





Il puntatore (2)

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- *2 data object* coinvolti.



Il puntatore (2)

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

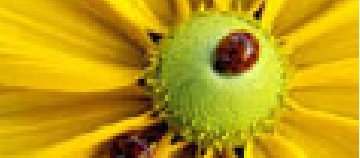
Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- 2 *data object* coinvolti.
- Il secondo può non avere un legame di nome o di valore.



Il puntatore (2)

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

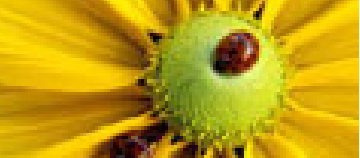
Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- 2 *data object* coinvolti.
- Il secondo può non avere un legame di nome o di valore.
- La deallocazione è necessaria, perché la modifica del legame di valore genera di solito dati non più accessibili per nome o riferimento.



Il puntatore (2)

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

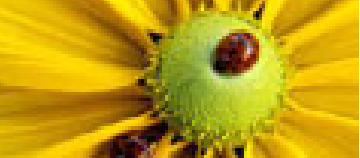
Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione

- 2 *data object* coinvolti.
- Il secondo può non avere un legame di nome o di valore.
- La deallocazione è necessaria, perché la modifica del legame di valore genera di solito dati non più accessibili per nome o riferimento.
- Alcuni linguaggi possiedono meccanismi di recupero automatico di memoria (*garbage collector*).



Data object in C

Inserire slide “Richiami di C”

Modello imperativo

Data Object e legami

Data Object: un altro modo di astrarre l'implementazione dei dati

Legami

Modifiche di legami

Esempio 1

Esempio 2

Esempio 3

Il puntatore (1)

Il puntatore (2)

Data object in C

Legami di tipo

Blocchi di istruzioni

Legami di locazione



Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

Legami di tipo



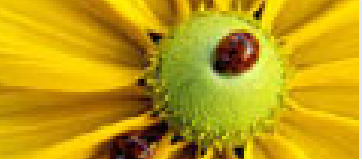
Legame di tipo

- Per definizione è correlato al legame di valore (ad es. il tipo di una variabile e il tipo del suo valore devono corrispondere).



Legame di tipo

- Per definizione è correlato al legame di valore (ad es. il tipo di una variabile e il tipo del suo valore devono corrispondere).
- Ogni volta che il legame di valore viene modificato, occorrerebbe controllare (type checking) la consistenza con il legame di tipo.



Legame di tipo

- Per definizione è correlato al legame di valore (ad es. il tipo di una variabile e il tipo del suo valore devono corrispondere).
- Ogni volta che il legame di valore viene modificato, occorrerebbe controllare (type checking) la consistenza con il legame di tipo.
- Un linguaggio è *dinamicamente tipizzato* se il legame (e le variazioni di legame) e di conseguenza anche il controllo di consistenza (se avviene) avvengono durante l'esecuzione.



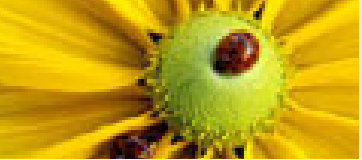
Legame di tipo

- Per definizione è correlato al legame di valore (ad es. il tipo di una variabile e il tipo del suo valore devono corrispondere).
- Ogni volta che il legame di valore viene modificato, occorrerebbe controllare (type checking) la consistenza con il legame di tipo.
- Un linguaggio è *dinamicamente tipizzato* se il legame (e le variazioni di legame) e di conseguenza anche il controllo di consistenza (se avviene) avvengono durante l'esecuzione.

Esempio: nei linguaggi di scripting e Python

```
x=1; ... x= "abc";
```

Inizialmente il tipo di x è numerico, poi è stringa (il legame di tipo cambia in seguito ad un cambio del legame di valore).



Legame di tipo

- Per definizione è correlato al legame di valore (ad es. il tipo di una variabile e il tipo del suo valore devono corrispondere).
- Ogni volta che il legame di valore viene modificato, occorrerebbe controllare (type checking) la consistenza con il legame di tipo.
- Un linguaggio è *dinamicamente tipizzato* se il legame (e le variazioni di legame) e di conseguenza anche il controllo di consistenza (se avviene) avvengono durante l'esecuzione.

Esempio: nei linguaggi di scripting e Python

```
x=1; ... x= "abc";
```

Inizialmente il tipo di x è numerico, poi è stringa (il legame di tipo cambia in seguito ad un cambio del legame di valore).

- Un linguaggio è *staticamente tipizzato* se il legame avviene durante la compilazione; in questo caso il controllo di consistenza (se avviene) può avvenire in entrambe le fasi.



Type checking (1)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

È il meccanismo di controllo di consistenza della coppia dei legami valore-tipo.



Type checking (1)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

È il meccanismo di controllo di consistenza della coppia dei legami valore-tipo.

Può avvenire: a) durante la compilazione, b) durante l'esecuzione, c) per nulla.



Type checking (1)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

È il meccanismo di controllo di consistenza della coppia dei legami valore-tipo.

Può avvenire: a) durante la compilazione, b) durante l'esecuzione, c) per nulla.

- Un linguaggio è *fortemente tipizzato* se il controllo di consistenza avviene sempre



Type checking (1)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre
approssimazioni del
controllo di tipi

Blocchi di istruzioni

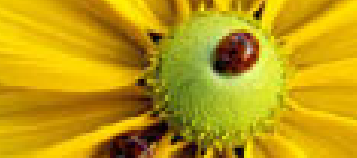
Legami di locazione

È il meccanismo di controllo di consistenza della coppia dei legami valore-tipo.

Può avvenire: a) durante la compilazione, b) durante l'esecuzione, c) per nulla.

■ Un linguaggio è *fortemente tipizzato* se il controllo di consistenza avviene sempre

◆ Java è fortemente tipizzato (vedremo poi).



Type checking (1)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre
approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

È il meccanismo di controllo di consistenza della coppia dei legami valore-tipo.

Può avvenire: a) durante la compilazione, b) durante l'esecuzione, c) per nulla.

- Un linguaggio è *fortemente tipizzato* se il controllo di consistenza avviene sempre
 - ◆ Java è fortemente tipizzato (vedremo poi).
 - ◆ Pascal è quasi fortemente tipizzato (una sola eccezione di assenza di controllo: i record con varianti).



Type checking (2)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

- Un linguaggio è *debolmente tipizzato* se il controllo di consistenza può non avvenire affatto in numerosi casi.



Type checking (2)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

- Un linguaggio è *debolmente tipizzato* se il controllo di consistenza può non avvenire affatto in numerosi casi.
 - ◆ C è debolmente tipizzato:
 - (1) supporta le operazioni di *casting* (o *coercizione di tipo*), che consentono di forzare, in esecuzione, l'interpretazione di un qualunque valore secondo un qualunque tipo;



Type checking (2)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

- Un linguaggio è *debolmente tipizzato* se il controllo di consistenza può non avvenire affatto in numerosi casi.
 - ◆ C è debolmente tipizzato:
 - (1) supporta le operazioni di *casting* (o *coercizione di tipo*), che consentono di forzare, in esecuzione, l'interpretazione di un qualunque valore secondo un qualunque tipo;
 - (2) supporta conversioni implicite (senza cast) tra tutti i tipi di puntatori;



Type checking (2)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

- Un linguaggio è *debolmente tipizzato* se il controllo di consistenza può non avvenire affatto in numerosi casi.
 - ◆ C è debolmente tipizzato:
 - (1) supporta le operazioni di *casting* (o *coercizione di tipo*), che consentono di forzare, in esecuzione, l'interpretazione di un qualunque valore secondo un qualunque tipo;
 - (2) supporta conversioni implicite (senza cast) tra tutti i tipi di puntatori;
 - (3) supporta le *unioni*, che sovrappongono la locazione di variabili di tipo diverso, senza controllare se il valore corrisponde al tipo con cui viene usato



Esempio di uso sbagliato di union non segnalato

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

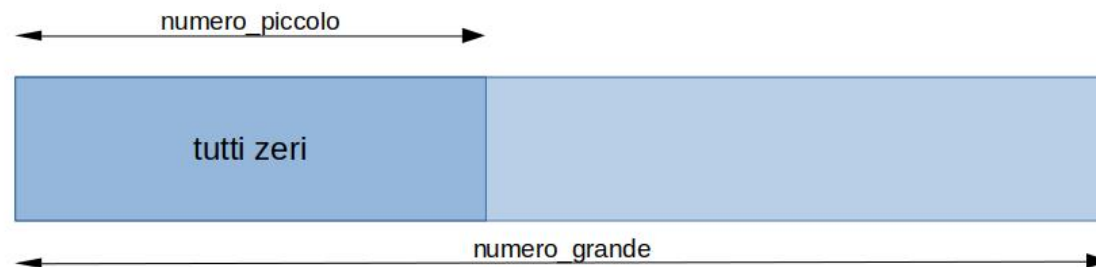
approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

```
union numero
{
    int numero_piccolo;
    double numero_grande;
} numeri;

void main() {
    numeri.numero_grande = 3.0;
    printf("%d\n", numeri.numero_piccolo);
}
```



- si inserisce un double ma si legge come fosse un int
- viene stampato '0'
- il compilatore non segnala l'errore



Il linguaggio perfetto (1)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

Sarebbe bello se esistesse un linguaggio Turing completo, in cui il type checking avvenisse completamente durante la compilazione e *in cui il compilatore non generasse più errori del necessario* – Linguaggio Perfetto (LP).

- LP, se esistesse, sarebbe staticamente e fortemente tipizzato (il più “forte” di tutti i fortemente tipizzati).



Il linguaggio perfetto (1)

Modello imperativo

Data Object e
legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso
sbagliato di union
non segnalato

Linguaggio perfetto

Linguaggio perfetto

Altre

approssimazioni del
controllo di tipi

Blocchi di istruzioni

Legami di locazione

Sarebbe bello se esistesse un linguaggio Turing completo, in cui il type checking avvenisse completamente durante la compilazione e *in cui il compilatore non generasse più errori del necessario* – Linguaggio Perfetto (LP).

- LP, se esistesse, sarebbe staticamente e fortemente tipizzato (il più “forte” di tutti i fortemente tipizzati).
- LP non può esistere.



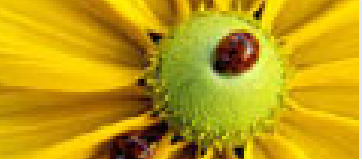
Il linguaggio perfetto (2)

- Se LP esistesse, allora il compilatore, per essere capace di decidere la correttezza dell'ultima assegnazione in:

```
int x;  
P;  
x = "pippo";
```

dove P è un generico programma, dovrebbe essere capace di decidere la terminazione di P (l'istruzione errata viene eseguita se e solo se la chiamata a P termina)

quindi LP non sarebbe Turing completo, contro l'ipotesi.



Il linguaggio perfetto (2)

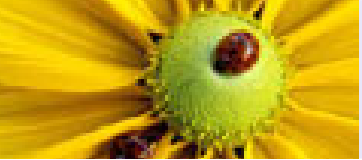
- Se LP esistesse, allora il compilatore, per essere capace di decidere la correttezza dell'ultima assegnazione in:

```
int x;  
P;  
x = "pippo";
```

dove P è un generico programma, dovrebbe essere capace di decidere la terminazione di P (l'istruzione errata viene eseguita se e solo se la chiamata a P termina)

quindi LP non sarebbe Turing completo, contro l'ipotesi.

- I compilatori, in situazioni come la precedente, ignorano se una istruzione viene eseguita o meno e segnalano comunque un errore nell'ultima linea.



Il linguaggio perfetto (2)

- Se LP esistesse, allora il compilatore, per essere capace di decidere la correttezza dell'ultima assegnazione in:

```
int x;  
P;  
x = "pippo";
```

dove P è un generico programma, dovrebbe essere capace di decidere la terminazione di P (l'istruzione errata viene eseguita se e solo se la chiamata a P termina)

quindi LP non sarebbe Turing completo, contro l'ipotesi.

- I compilatori, in situazioni come la precedente, ignorano se una istruzione viene eseguita o meno e segnalano comunque un errore nell'ultima linea.
Per esempio, questo programma Pascal non compila anche se l'istruzione errata non verrebbe eseguita:

```
program wrong (input, output);  
var i: integer;  
begin  
    if false then i:= 3.14;  
        else i:= 0;  
end.
```




Altre approssimazioni del controllo di tipi

Modello imperativo

Data Object e legami

Legami di tipo

Legame di tipo

Type checking (1)

Type checking (2)

Esempio di uso sbagliato di union non segnalato

Linguaggio perfetto

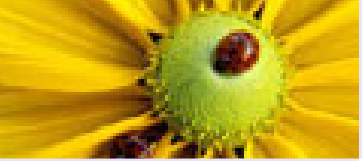
Linguaggio perfetto

Altre approssimazioni del controllo di tipi

Blocchi di istruzioni

Legami di locazione

- Alcuni controlli di tipo sono impossibili a tempo di compilazione
 - ◆ ad esempio la correttezza dei “down cast” in Java, che vedremo più avanti
 - Quindi la strategia per i linguaggi fortemente e staticamente tipati è
 - ◆ fare quanti più controlli possibile a tempo di compilazione
 - ◆ eseguire i rimanenti a tempo di esecuzione
- perchè prima si scoprono gli errori e più il testing diventa economico



Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Blocchi di istruzioni



Scope

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Si definisce *scope* o *ambito di validità* di un legame, l'insieme degli enunciati del programma in cui quel legame è valido.

I linguaggi moderni limitano lo scope usando *blocchi* di codice

I blocchi possono anche prendere la forma di procedure, funzioni, classi, etc.



Necessità dei blocchi

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Istruzioni raggruppate in blocchi per meglio definire:

- ambito delle strutture di controllo (*Dove finisce il corpo di questo ciclo?*)



Necessità dei blocchi

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Istruzioni raggruppate in blocchi per meglio definire:

- ambito delle strutture di controllo (*Dove finisce il corpo di questo ciclo?*)
- ambito di una procedura (*Dove finisce questa procedura?*)



Necessità dei blocchi

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Istruzioni raggruppate in blocchi per meglio definire:

- ambito delle strutture di controllo (*Dove finisce il corpo di questo ciclo?*)
- ambito di una procedura (*Dove finisce questa procedura?*)
- unità di compilazione separata;



Necessità dei blocchi

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Istruzioni raggruppate in blocchi per meglio definire:

- ambito delle strutture di controllo (*Dove finisce il corpo di questo ciclo?*)
- ambito di una procedura (*Dove finisce questa procedura?*)
- unità di compilazione separata;
- ambito di validità dei legami di nome (*Dove è visibile questo nome?*)



Definizioni

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

A scopo didattico, introduciamo un semplice pseudo-linguaggio di *blocchi di codice*.

Un blocco contiene due parti:

- una sezione di dichiarazione di nomi (cioè, variabili);



Definizioni

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

A scopo didattico, introduciamo un semplice pseudo-linguaggio di *blocchi di codice*.

Un blocco contiene due parti:

- una sezione di dichiarazione di nomi (cioè, variabili);
- una sezione che comprende gli enunciati sui quali hanno validità quei legami di nome.



Definizioni

[Modello imperativo](#)

[Data Object e legami](#)

[Legami di tipo](#)

[Blocchi di istruzioni](#)

[Scope](#)

[Necessità dei blocchi](#)

[Definizioni](#)

[Ambito di . . .](#)

[Esempio di scoping statico](#)

[Mascheramento](#)

[Mascheramento](#)

[Legami di locazione](#)

A scopo didattico, introduciamo un semplice pseudo-linguaggio di *blocchi di codice*.

Un blocco contiene due parti:

- una sezione di dichiarazione di nomi (cioè, variabili);
- una sezione che comprende gli enunciati sui quali hanno validità quei legami di nome.

Esempio:

```
...  
BLOCK A;  
    DECLARE I;  
BEGIN A  
    ...                {I DEL BLOCCO A}  
END A;  
...
```



Ambito di validità di legami

Essenzialmente due tipi di ambito di validità (scoping):

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione



Ambito di validità di legami

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Essenzialmente due tipi di ambito di validità (scoping):

Scoping statico o lessicale

Blocchi annidati vedono e usano i legami dei blocchi più esterni (*legami non locali*) e possono aggiungere legami locali o sovrapporne di nuovi.



Ambito di validità di legami

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Essenzialmente due tipi di ambito di validità (scoping):

Scoping statico o lessicale

Blocchi annidati vedono e usano i legami dei blocchi più esterni (*legami non locali*) e possono aggiungere legami locali o sovrapporne di nuovi.

Scoping dinamico

Concetto qui esaminato solo in relazione ai blocchi annidati, ma che assume il proprio senso maggiore quando vi sono procedure chiamanti e chiamate. In questo caso la procedura chiamata vede e usa i legami visti e usati dalla procedura chiamante.



Ambito di validità di legami

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

Essenzialmente due tipi di ambito di validità (scoping):

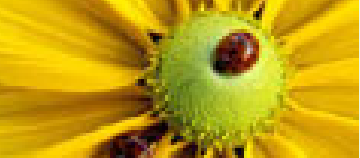
Scoping statico o lessicale

Blocchi annidati vedono e usano i legami dei blocchi più esterni (*legami non locali*) e possono aggiungere legami locali o sovrapporne di nuovi.

Scoping dinamico

Concetto qui esaminato solo in relazione ai blocchi annidati, ma che assume il proprio senso maggiore quando vi sono procedure chiamanti e chiamate. In questo caso la procedura chiamata vede e usa i legami visti e usati dalla procedura chiamante.

Esamineremo tutti i dettagli nella prossima lezione.



Esempio di scoping statico

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

**Esempio di scoping
statico**

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X;  
BEGIN P  
  . . .           {X da P}
```



Esempio di scoping statico

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

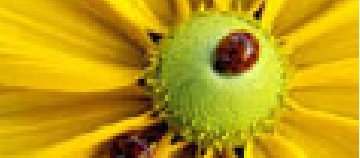
**Esempio di scoping
statico**

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X;  
BEGIN P  
  ...                {X da P}  
  BLOCK A;  
    DECLARE Y;  
  BEGIN A  
    ...                {X da P, Y da A}
```

Esempio di scoping statico

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

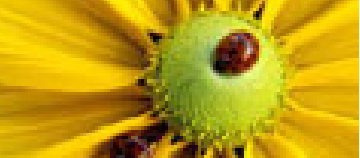
**Esempio di scoping
statico**

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X;  
BEGIN P  
  ...                               {X da P}  
  BLOCK A;  
    DECLARE Y;  
  BEGIN A  
    ...                             {X da P, Y da A}  
    BLOCK B;  
      DECLARE Z;  
    BEGIN B  
      ...                           {X da P, Y da A, Z da B}
```



Esempio di scoping statico

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

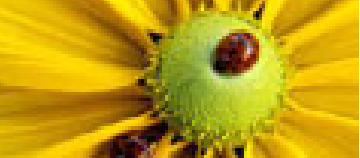
**Esempio di scoping
statico**

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X;  
BEGIN P  
  ...                               {X da P}  
  BLOCK A;  
    DECLARE Y;  
  BEGIN A  
    ...                             {X da P, Y da A}  
    BLOCK B;  
      DECLARE Z;  
    BEGIN B  
      ...                           {X da P, Y da A, Z da B}  
    END B;  
    ...                             {X da P, Y da A}
```



Esempio di scoping statico

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

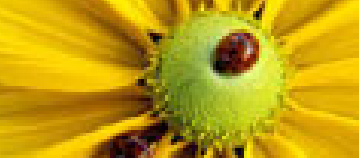
**Esempio di scoping
statico**

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X;  
BEGIN P  
  ... {X da P}  
  BLOCK A;  
    DECLARE Y;  
  BEGIN A  
    ... {X da P, Y da A}  
    BLOCK B;  
      DECLARE Z;  
    BEGIN B  
      ... {X da P, Y da A, Z da B}  
    END B;  
    ... {X da P, Y da A}  
  END A;  
  ... {X da P}
```



Esempio di scoping statico

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

**Esempio di scoping
statico**

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X;  
BEGIN P  
  ... {X da P}  
  BLOCK A;  
    DECLARE Y;  
  BEGIN A  
    ... {X da P, Y da A}  
    BLOCK B;  
      DECLARE Z;  
    BEGIN B  
      ... {X da P, Y da A, Z da B}  
    END B;  
    ... {X da P, Y da A}  
  END A;  
  ... {X da P}  
  BLOCK C;  
    DECLARE Z;  
  START C  
    ... {X da P, Z da C}
```



Esempio di scoping statico

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

**Esempio di scoping
statico**

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X;  
  BEGIN P  
    ... {X da P}  
    BLOCK A;  
      DECLARE Y;  
      BEGIN A  
        ... {X da P, Y da A}  
        BLOCK B;  
          DECLARE Z;  
          BEGIN B  
            ... {X da P, Y da A, Z da B}  
          END B;  
          ... {X da P, Y da A}  
        END A;  
        ... {X da P}  
      BLOCK C;  
        DECLARE Z;  
        START C  
          ... {X da P, Z da C}  
        END C;  
        ... {X da P}  
      END P;
```



Mascheramento

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di ...

Esempio di scoping
statico

Mascheramento

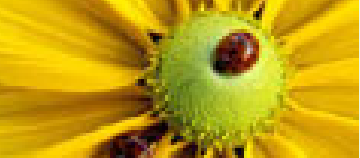
Mascheramento

Legami di locazione

- Si parla di *mascheramento* o *shadowing* quando un blocco interno ridefinisce un nome che era già definito in un blocco esterno
- In tal caso, la nuova definizione nasconde (maschera) la vecchia, finché non si esce dal blocco interno

In C:

```
void f(int n) {  
    int i = 0;  
    {  
        int n = 0; // maschera n  
        n++;      // incrementa la nuova n  
    }  
    n--;          // decrementa la vecchia n (il par. formale)  
}
```



Mascheramento

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

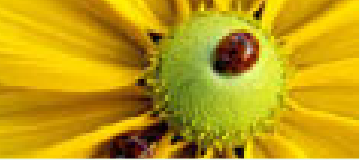
Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X,Y;  
BEGIN P  
  . . .                {X e Y da P}
```



Mascheramento

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

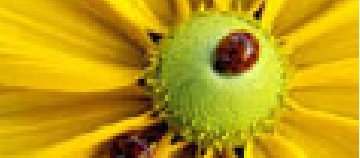
Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X,Y;  
BEGIN P  
  ...                {X e Y da P}  
  BLOCK A;  
    DECLARE X,Z;  
  BEGIN A  
    ...              {X e Z da A, Y da P}
```

Mascheramento

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Scope

Necessità dei blocchi

Definizioni

Ambito di . . .

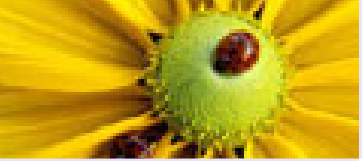
Esempio di scoping
statico

Mascheramento

Mascheramento

Legami di locazione

```
PROGRAM P;  
  DECLARE X,Y;  
BEGIN P  
  ...           {X e Y da P}  
  BLOCK A;  
    DECLARE X,Z;  
  BEGIN A  
    ...           {X e Z da A, Y da P}  
  END A;  
  ...           {X e Y da P}  
END P;
```



Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

Legami di locazione



Legami di locazione e scope

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

- Stabilire un legame di locazione si chiama *allocare memoria*
- Se una variabile è in scope, deve avere un legame di locazione
- Non vale il viceversa: un legame di locazione può essere attivo mentre la variabile non è in scope (ad es., per mascheramento)
- Quindi, lo scope (ambito di visibilità) di una variabile è un **sottoinsieme** dell'ambito di validità del suo legame di locazione



Tecniche di allocazione di memoria

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione
Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

- **Statica:** l'indirizzo viene fissato a load-time e tale allocazione vale per tutta l'esecuzione
- **Dinamica**
 - ◆ Su stack: l'indirizzo viene fissato a runtime su richiesta (ad es., all'inizio di un blocco) e viene rilasciato al termine del blocco corrente
 - ◆ Su heap: l'indirizzo viene fissato a runtime su richiesta e rilasciato su richiesta oppure automaticamente quando non più utilizzato (garbage collection)



Allocazione statica di memoria

Si dice *allocazione statica di memoria* quando il legame di locazione è fissato e costante al tempo di caricamento (load-time)

Modello imperativo

Data Object e legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio



Allocazione statica di memoria

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

Si dice *allocazione statica di memoria* quando il legame di locazione è fissato e costante al tempo di caricamento (load-time)
Tipico delle variabili globali (se previste dal linguaggio)



Allocazione statica di memoria

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

Si dice *allocazione statica di memoria* quando il legame di locazione è fissato e costante al tempo di caricamento (load-time)
Tipico delle variabili globali (se previste dal linguaggio)
In C:

```
int g; // alloc. statica e scope globale

void f(int n) {
    static int a; // alloc. statica e scope locale
    int i;
    ...
}
```



Allocazione dinamica di memoria su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

- Si dice *allocazione dinamica di memoria tramite stack* quando il legame di locazione è creato all'inizio dell'esecuzione di un blocco e viene rilasciato automaticamente a fine blocco



Allocazione dinamica di memoria su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

- Si dice *allocazione dinamica di memoria tramite stack* quando il legame di locazione è creato all'inizio dell'esecuzione di un blocco e viene rilasciato automaticamente a fine blocco
- Essa è realizzata attraverso il *Record di Attivazione (RdA)* di un blocco



Allocazione dinamica di memoria su stack

Modello imperativo

Data Object e legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

- Si dice *allocazione dinamica di memoria tramite stack* quando il legame di locazione è creato all'inizio dell'esecuzione di un blocco e viene rilasciato automaticamente a fine blocco
- Essa è realizzata attraverso il *Record di Attivazione (RdA)* di un blocco
- L'RdA contiene tutte le informazioni necessarie all'esecuzione del blocco e alla prosecuzione dell'esecuzione dopo la fine del blocco



L'RdA di un blocco anonimo

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

- Contenuto dell'RdA nel caso di un blocco anonimo o *in-line*:
 - ◆ Puntatore di catena dinamica (link all'RdA precedente)
 - ◆ Ambiente locale (variabili locali e spazio per risultati intermedi)



L'RdA di un blocco anonimo

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

- Contenuto dell'RdA nel caso di un blocco anonimo o *in-line*:
 - ◆ Puntatore di catena dinamica (link all'RdA precedente)
 - ◆ Ambiente locale (variabili locali e spazio per risultati intermedi)
- Quando invece il blocco è più complesso (ad es., è una procedura e può essere invocato), l'RdA è più complesso, come si vedrà in seguito



Alloc. su stack in C

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

In C:

```
int g;

void f(int n) { // n: alloc. dinamica su stack e scope locale (
    static int a;
    int i;      // i: alloc. dinamica su stack e scope locale (
    {
        int j;  // j: alloc. dinamica su stack e scope locale (
    }
    ...
}
```



Stack di esecuzione

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

In ogni momento dell'esecuzione lo *stack (o pila) di esecuzione* contiene gli RdA “attivi” :



Stack di esecuzione

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

In ogni momento dell'esecuzione lo *stack (o pila) di esecuzione* contiene gli RdA “attivi” :

1. il top dello stack contiene l'RdA del blocco correntemente in esecuzione;



Stack di esecuzione

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

In ogni momento dell'esecuzione lo *stack (o pila) di esecuzione* contiene gli RdA “attivi” :

1. il top dello stack contiene l'RdA del blocco correntemente in esecuzione;
2. ogni volta che si entra in un blocco, l'RdA del blocco viene creato e posto sullo stack (push);



Stack di esecuzione

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

In ogni momento dell'esecuzione lo *stack (o pila) di esecuzione* contiene gli RdA “attivi” :

1. il top dello stack contiene l'RdA del blocco correntemente in esecuzione;
2. ogni volta che si entra in un blocco, l'RdA del blocco viene creato e posto sullo stack (push);
3. ogni volta che si esce da un blocco, viene eliminato l'RdA in cima allo stack (pop)



Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

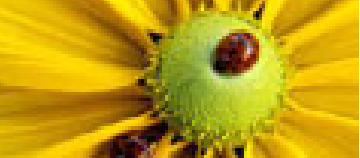
Esempio

All'ingresso in P (supponendo che anche I e J siano allocate sullo stack)

Contenuto dello
stack di attivazione

I, J	P
null	

```
PROGRAM P;  
  DECLARE I,J;  
BEGIN P  
  BLOCK A;  
    DECLARE I,K;  
  BEGIN A  
    BLOCK B;  
      DECLARE I,L;  
    BEGIN B  
      ... {I e L da B, K da A, J da P}  
    END B;  
      ... {I e K da A, J da P}  
    END A;  
  BLOCK C;  
    DECLARE I,N;  
  BEGIN C  
    ... {I e N da C, J da P}  
  END C;  
  ... {I e J da P}  
END P;
```



Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

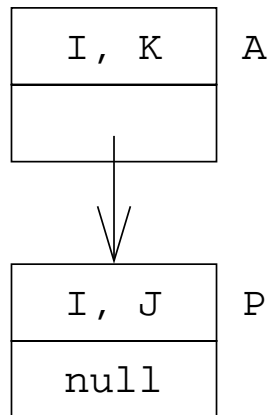
Alloc. su stack in C

Stack di esecuzione

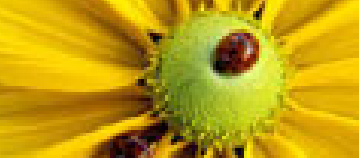
Esempio

All'ingresso in A

Contenuto dello
stack di attivazione



```
PROGRAM P;  
  DECLARE I, J;  
  BEGIN P  
    BLOCK A;  
      DECLARE I, K;  
      BEGIN A  
        BLOCK B;  
          DECLARE I, L;  
          BEGIN B  
            ... {I e L da B, K da A, J da P}  
          END B;  
          ... {I e K da A, J da P}  
        END A;  
        BLOCK C;  
          DECLARE I, N;  
          BEGIN C  
            ... {I e N da C, J da P}  
          END C;  
          ... {I e J da P}  
        ...  
      END P;
```



Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

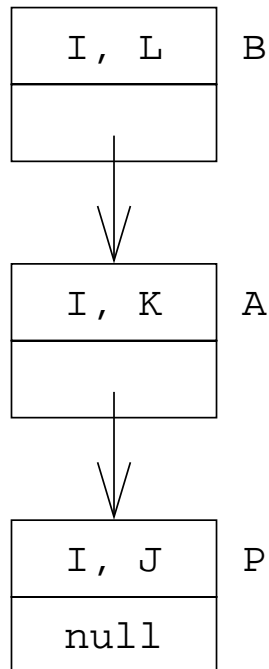
Alloc. su stack in C

Stack di esecuzione

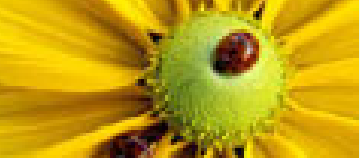
Esempio

All'ingresso in B

Contenuto dello
stack di attivazione



```
PROGRAM P;  
  DECLARE I,J;  
BEGIN P  
  BLOCK A;  
    DECLARE I,K;  
  BEGIN A  
    BLOCK B;  
      DECLARE I,L;  
    BEGIN B  
      ... {I e L da B, K da A, J da P}  
    END B;  
    ... {I e K da A, J da P}  
  END A;  
  BLOCK C;  
    DECLARE I,N;  
  BEGIN C  
    ... {I e N da C, J da P}  
  END C;  
  ... {I e J da P}  
END P;
```



Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

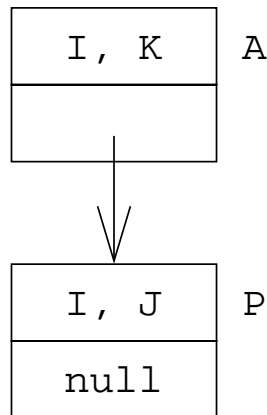
Alloc. su stack in C

Stack di esecuzione

Esempio

All'uscita da B

Contenuto dello stack di attivazione



```
PROGRAM P;  
  DECLARE I,J;  
BEGIN P  
  BLOCK A;  
    DECLARE I,K;  
  BEGIN A  
    BLOCK B;  
      DECLARE I,L;  
    BEGIN B  
      ... {I e L da B, K da A, J da P}  
    END B;  
    ... {I e K da A, J da P}  
  END A;  
  BLOCK C;  
    DECLARE I,N;  
  BEGIN C  
    ... {I e N da C, J da P}  
  END C;  
  ... {I e J da P}  
END P;
```



Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

Esempio

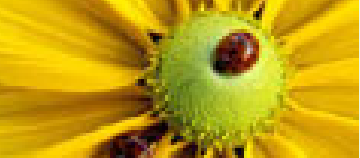
All'uscita da A

Contenuto dello
stack di attivazione

I, J	P
null	

```

PROGRAM P;
  DECLARE I,J;
BEGIN P
  BLOCK A;
    DECLARE I,K;
  BEGIN A
    BLOCK B;
      DECLARE I,L;
    BEGIN B
      ... {I e L da B, K da A, J da P}
    END B;
      ... {I e K da A, J da P}
    END A;
  BLOCK C;
    DECLARE I,N;
  BEGIN C
    ... {I e N da C, J da P}
  END C;
  ... {I e J da P}
END P;
    
```



Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

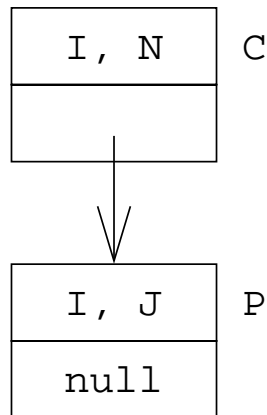
Alloc. su stack in C

Stack di esecuzione

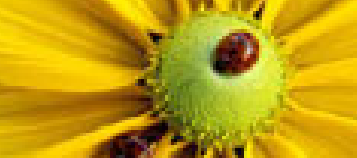
Esempio

All'ingresso in C

Contenuto dello
stack di attivazione



```
PROGRAM P;  
  DECLARE I, J;  
BEGIN P  
  BLOCK A;  
    DECLARE I, K;  
  BEGIN A  
    BLOCK B;  
      DECLARE I, L;  
    BEGIN B  
      ... {I e L da B, K da A, J da P}  
    END B;  
    ... {I e K da A, J da P}  
  END A;  
  BLOCK C;  
    DECLARE I, N;  
  BEGIN C  
    ... {I e N da C, J da P}  
  END C;  
  ... {I e J da P}  
END P;
```



Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

Stack di esecuzione

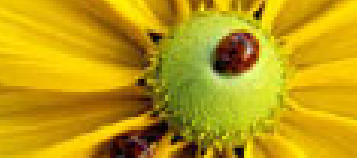
Esempio

All'uscita da C

Contenuto dello
stack di attivazione

I, J	P
null	

```
PROGRAM P;  
  DECLARE I,J;  
BEGIN P  
  BLOCK A;  
    DECLARE I,K;  
  BEGIN A  
    BLOCK B;  
      DECLARE I,L;  
    BEGIN B  
      ... {I e L da B, K da A, J da P}  
    END B;  
      ... {I e K da A, J da P}  
    END A;  
  BLOCK C;  
    DECLARE I,N;  
  BEGIN C  
    ... {I e N da C, J da P}  
  END C;  
  ... {I e J da P}  
END P;
```

Esempio di allocazione dinamica su stack

Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legami di locazione

Legami di locazione
e scope

Allocazione

Alloc. statica

Alloc. dinamica

RdA di blocco

Alloc. su stack in C

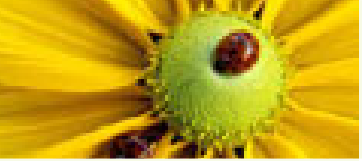
Stack di esecuzione

Esempio

All'uscita da P

Contenuto dello
stack di attivazione

```
PROGRAM P;  
  DECLARE I,J;  
BEGIN P  
  BLOCK A;  
    DECLARE I,K;  
  BEGIN A  
    BLOCK B;  
      DECLARE I,L;  
    BEGIN B  
      ... {I e L da B, K da A, J da P}  
    END B;  
      ... {I e K da A, J da P}  
    END A;  
  BLOCK C;  
    DECLARE I,N;  
  BEGIN C  
    ... {I e N da C, J da P}  
  END C;  
  ... {I e J da P}  
END P;
```



Modello imperativo

Data Object e
legami

Legami di tipo

Blocchi di istruzioni

Legame di locazione

Esercizi

Esercizio 1

Esercizio 2

Bibliografia

Esercizi



Esercizio 1

Si considerino i seguenti blocchi annidati. Per ciascun blocco, determinare i legami di ogni variabile. Evidenziare le variabili locali e non locali.

```
PROGRAM P;  
  BLOCK B1;  
    DECLARE A,B,C;  
  BEGIN B1  
    BLOCK B2;  
      DECLARE C,D;  
    BEGIN B2  
      BLOCK B3;  
        DECLARE B,D,F;  
      BEGIN  
        ...  
      END B3;  
      ...  
    END B2;  
    BLOCK B4;  
      DECLARE B,C,D;  
    BEGIN B4  
      ...  
    END B4;  
    ...  
  END B1;  
END P;
```



Esercizio 2

Tracciare il contenuto dello stack di esecuzione durante l'esecuzione del seguente pseudo-programma.

```
PROGRAM P;  
  DECLARE X,Y;  
BEGIN P  
  BLOCK A;  
    DECLARE X,Y,Z;  
  BEGIN A  
    BLOCK B;  
      DECLARE Y;  
    BEGIN B  
      BLOCK C;  
        DECLARE X,Y;  
      BEGIN C  
        ...  
      END C;  
    BLOCK D;  
      DECLARE Z;  
    BEGIN D
```

```
        ...  
      END D;  
    ...  
  END B;  
END A;  
BLOCK E;  
  DECLARE Z;  
BEGIN E  
  BLOCK F;  
    DECLARE X;  
  BEGIN F  
    ...  
  END F;  
  ...  
END E;  
...  
END P;
```



Bibliografia

[Modello imperativo](#)

[Data Object e legami](#)

[Legami di tipo](#)

[Blocchi di istruzioni](#)

[Legami di locazione](#)

[Esercizi](#)

[Esercizio 1](#)

[Esercizio 2](#)

[Bibliografia](#)

Capitoli 6 (“I nomi e l’ambiente”) e 7 (“La gestione della memoria”) di *Linguaggi di Programmazione, principi e paradigmi*, di Gabbrielli e Martini (2a edizione)