

 UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Corso di Laurea in Informatica (*Track B*) - A.A. 2018/2019

Laboratorio di Informatica

Puntatori

(Parte 1)

docente: Veronica Rossano

Veronica.rossano@uniba.it

Slides a cura del dott. Corrado Mancuso

Slides ispirate ai contenuti proposti dal prof. Corrado Mencar, Grazie.

Variabili

- Cosa sono?
 - Si definisce «variabile» una particolare **porzione di memoria** destinata a contenere dei valori **acquisiti, elaborati o prodotti da un algoritmo**
- Come descrive una variabile?

Slides a cura del dott. Corrado Mancuso

10/04/19 Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019 3

Variabili

- Cosa sono?

Slides a cura del dott. Corrado Mancuso

10/04/19 Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019 2

Variabili

- Cosa sono?
 - Si definisce «variabile» una particolare **porzione di memoria** destinata a contenere dei valori **acquisiti, elaborati o prodotti da un algoritmo**
- Come descrive una variabile?
 - Nome
 - Tipo
 - Indirizzo di memoria
- Nelle variabili tradizionali, i primi due sono **esplicitamente** dichiarati, il terzo è «nascosto»
 - **Es.** `int i = 0;`
 - **Es.** `float a = 8.0;`

Slides a cura del dott. Corrado Mancuso

10/04/19 Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019 4

Variabili

- Cosa sono?
 - Si definisce «variabile» una particolare **porzione di memoria** destinata a contenere dei valori **acquisiti, elaborati o prodotti da un algoritmo**
- Come descrive una variabile?
 - Nome
 - Tipo
 - Indirizzo di memoria
- Nelle variabili tradizionali, i primi due sono **esplicitamente** dichiarati, il terzo è «nascosto»
 - Es. int i = 0;
 - Es. float a = 8.0;

Come facciamo a recuperare
l'indirizzo di memoria di una
variabile?

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

5

Slide a cura del dott. Claudio Muto

Variabili (cont.)

- Come facciamo a risalire **all'indirizzo di memoria** di una variabile?

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

6

Slide a cura del dott. Claudio Muto

Variabili (cont.)

- Come facciamo a risalire **all'indirizzo di memoria** di una variabile?
- Il C mette a disposizione **l'operatore «&» (operatore indirizzo)**
 - Si usa **a fianco del nome di una variabile**
 - Restituisce l'**indirizzo di memoria** di quella variabile

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

7

Slide a cura del dott. Claudio Muto

Variabili (cont.)

- Come facciamo a risalire **all'indirizzo di memoria** di una variabile?
- Il C mette a disposizione **l'operatore «&&» (operatore indirizzo)**
 - Si usa **a fianco del nome di una variabile**
 - Restituisce l'**indirizzo di memoria** di quella variabile

```

1 #include <stdio.h>
2
3 int main() {
4     int a = 5; // variabili intere
5     int b = 3;
6
7     // stampo valore e indirizzo
8     printf("a=%d \t b=%d \n", a, b);
9     printf("&a=%X \t &b=%X \n", &a, &b);
10 }
```

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

8

Slide a cura del dott. Claudio Muto

Variabili (cont.)

- Come facciamo a risalire **all'indirizzo di memoria** di una variabile?
- Il C mette a disposizione **l'operatore &&** (**operatore indirizzo**)
 - Si usa **a fianco del nome di una variabile**
 - Restituisce **l'indirizzo di memoria** di quella variabile

```

1 #include <stdio.h>
2
3 int main() {
4     int a = 5; // variabili intere
5     int b = 3;
6
7     // stampo valore e indirizzo
8     printf("a=%d \t\t b=%d \n", a, b);
9     printf("&a=%X \t &b=%X \n", &a, &b);
10 }
```

10/04/19

Veronica Rossano - Puntatori [Parte 1] Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

Slide a cura del dott. Claudio Musto

9

Variabili (cont.)

```

#include <stdio.h>

int main() {
    int a = 5; // variabili intere
    int b = 3;

    // stampo valore e indirizzo
    printf("a=%d \t\t b=%d \n", a, b);
    printf("&a=%X \t &b=%X \n", &a, &b);
}
```



10/04/19

Veronica Rossano - Puntatori [Parte 1] Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

Slide a cura del dott. Claudio Musto

10

Variabili (cont.)

```

#include <stdio.h>

int main() {
    int a = 5; // variabili intere
    int b = 3;

    // stampo valore e indirizzo
    printf("a=%d \t\t b=%d \n", a, b);
    printf("&a=%X \t &b=%X \n", &a, &b);
}
```



```

gcc version 4.6.3
:
a=5          b=3
&a=657D6F6C  &b=657D6F68
:
```

Slide a

10/04/19

Veronica Rossano - Puntatori [Parte 1] Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

11

Variabili (cont.)

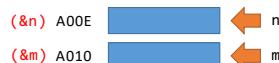
- Quando abbiamo già visto **l'operatore indirizzo**?

Slide a cura del dott. Claudio Musto

12

Variabili (cont.)

- Quando abbiamo già visto l'**operatore indirizzo**?
- Nei parametri della funzione **scanf()**
 - Es. **scanf("%d %d", &n, &m);**
 - In questo caso la **funzione risale all'indirizzo di memoria della variabile** e lo usa per memorizzarne il nuovo valore



Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

13

Variabili (cont.)

- Quando abbiamo già visto l'**operatore indirizzo**?
- Nei parametri della funzione **scanf()**
 - Es. **scanf("%d %d", &n, &m);**
 - In questo caso la **funzione risale all'indirizzo di memoria della variabile** e lo usa per memorizzarne il nuovo valore
 - Il **nome della variabile** è un «**alias**» (o un **nickname**) per riferirci a un indirizzo di memoria dell'area dati del programma

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

14

Variabili (cont.)

- Quando abbiamo già visto l'**operatore indirizzo**?
- Nei parametri della funzione **scanf()**
 - Es. **scanf("%d %d", &n, &m);**
 - In questo caso la **funzione risale all'indirizzo di memoria della variabile** e lo usa per memorizzarne il nuovo valore
 - Il **nome della variabile** è un «**alias**» (o un **nickname**) per riferirci a un indirizzo di memoria dell'area dati del programma
 - L'**operatore indirizzo** (insieme all'**operatore di indirezione**, che sarà introdotto successivamente) è di fondamentale importanza per l'uso dei puntatori

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

15

Puntatori

- Cosa sono?

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

16

Puntatori

- Cosa sono?
 - Una particolare tipologia di variabili
 - **Dov'è la particolarità?** Possono memorizzare solo **indirizzi di memoria**.
 - Normalmente le variabili contengono un **valore compatibile col tipo della variabile**.
 - Le variabili di tipo puntatore contengono un indirizzo di memoria

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

17

Slide a cura del dott. Claudio Musto

Puntatori

- Cosa sono?
 - Una particolare tipologia di variabili
 - **Dov'è la particolarità?** Possono memorizzare solo **indirizzi di memoria**.
 - Normalmente le variabili contengono un **valore compatibile col tipo della variabile**.
 - Le variabili di tipo puntatore contengono un indirizzo di memoria

A00E	5	→ Variabili tradizionali (<code>int n = 5</code>)
A010	c	→ Variabili tradizionali (<code>char c = 'c'</code>)
A012		
A014	8.0	→ Variabili tradizionali (<code>float x = 8.0</code>)

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

18

Slide a cura del dott. Claudio Musto

Puntatori

- Cosa sono?
 - Una particolare tipologia di variabili
 - **Dov'è la particolarità?** Possono memorizzare solo **indirizzi di memoria**.
 - Normalmente le variabili contengono un **valore compatibile col tipo della variabile**.
 - Le variabili di tipo puntatore contengono un indirizzo di memoria

A00E	5	→ Variabili tradizionali (<code>int n = 5</code>)
A010	c	→ Variabili tradizionali (<code>char c = 'c'</code>)
A012	A00E	→ Variabile puntatore
A014	8.0	→ Variabili tradizionali (<code>float x = 8.0</code>)

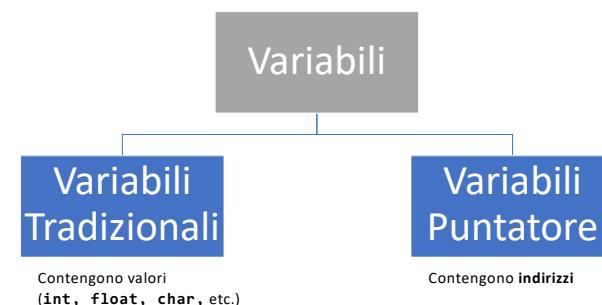
10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

19

Slide a cura del dott. Claudio Musto

Recap



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

20

Slide a cura del dott. Claudio Musto

Recap

```

graph TD
    Variabili[Variabili] --> VariabiliTradizionali[Variabili Tradizionali]
    Variabili --> VariabiliPuntatore[Variabili Puntatore]
    subgraph RedBox [ ]
        Questo schema non tiene in considerazione le variabili di tipo 'strutturato' o i tipi di dato definito dall'utente!
    end

```

Variabili

Variabili Tradizionali
Contengono valori
(`int`, `float`, `char`, etc.)

Variabili Puntatore
Contengono indirizzi

Slide a cura del dott. Claudio Musto

Variabili puntatore

- Come si dichiara?

10/04/19 Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019 22

Variabili puntatore

- Come si dichiara?
 - `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`

Slide a cura del dott. Claudio Musto

Variabili puntatore

- Come si dichiara?
 - `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`
 - L'asterisco può anche essere messo vicino al nome della variabile (`int *a`)
- Quali sono le caratteristiche di questa variabile? (es. `int *a`)
 - nome:**
 - tipo:**
 - valore:**
 - indirizzo:**

10/04/19 Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019 24

Variabili puntatore

- Come si dichiara?
 - `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`
- L'asterisco può anche essere messo vicino al nome della variabile (`int *a`)
- Quali sono le caratteristiche di questa variabile? (es. `int *a`)
 - **nome:** a
 - **tipo:** puntatore a intero (Si legge «a è un puntatore a un intero»)
 - **valore:** inizialmente casuale (← attenzione!)
 - **indirizzo:** definito dal compilatore

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

25

Slide a cura del dott. Claudio Musto

Variabili puntatore

- Come si dichiara?
 - `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`
- Le variabili di tipo puntatore hanno anche un tipo. **Cosa significa?**

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

26

Slide a cura del dott. Claudio Musto

Variabili puntatore

- Come si dichiara?
 - `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`
- Le variabili di tipo puntatore hanno anche un tipo. **Cosa significa?**
 - Il valore delle variabili di tipo puntatore deve essere un indirizzo
 - Quell'indirizzo deve contenere a sua volta un valore di quel tipo
 - Bisogna stare attenti: il compilatore spesso non restituisce errore. Gli errori si notano solo in fase di esecuzione

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

27

Slide a cura del dott. Claudio Musto

Variabili puntatore

- Come si dichiara?
 - `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`
- Le variabili di tipo puntatore hanno anche un tipo. **Cosa significa?**
 - Il valore delle variabili di tipo puntatore deve essere un indirizzo
 - Quell'indirizzo deve contenere a sua volta un valore di quel tipo
 - `int* a` → ad 'a' bisogna assegnare il valore di un indirizzo al cui interno deve essere memorizzato un valore intero!

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

28

Slide a cura del dott. Claudio Musto

Variabili puntatore - Esempio

- Come si dichiara?

- `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`

- Qual è un'assegnazione valida per la variabile `int* a` ?

A00E	5	→ Variabili tradizionali (<code>int n = 5</code>)
A010	c	→ Variabili tradizionali (<code>char c = 'c'</code>)
A012	????	→ Variabile puntatore (<code>int* a</code>)
A014	8.0	→ Variabili tradizionali (<code>float x = 8.0</code>)

Reminder
ad 'a' bisogna assegnare il valore di un indirizzo al cui interno deve essere memorizzato un valore intero!

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

29

Slide a cura del dott. Claudio Musto

Variabili puntatore - Esempio

- Come si dichiara?

- `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`

- Qual è un'assegnazione valida per la variabile `int* a` ?

A00E	5	→ Variabili tradizionali (<code>int n = 5</code>)
A010	c	→ Variabili tradizionali (<code>char c = 'c'</code>)
A012	A00E	→ Variabile puntatore (<code>int* a</code>)
A014	8.0	→ Variabili tradizionali (<code>float x = 8.0</code>)

Reminder
ad 'a' bisogna assegnare il valore di un indirizzo al cui interno deve essere memorizzato un valore intero!

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

30

Slide a cura del dott. Claudio Musto

Variabili puntatore - Esempio

- Come si dichiara?

- `<tipo_primitivo>* <nome_variabile>`
 - `int* a;`
 - `float* b;`
 - `char* c;`

- Qual è un'assegnazione valida per la variabile `int* a` ?

A00E	5	→ Variabili tradizionali (<code>int n = 5</code>)
A010	c	→ Variabili tradizionali (<code>char c = 'c'</code>)
A012	A00E	→ Variabile puntatore (<code>int* a</code>)
A014	8.0	→ Variabili tradizionali (<code>float x = 8.0</code>)

L'unica assegnazione valida è A00E perché la variabile deve 'puntare' a un indirizzo di memoria al cui interno è stato memorizzato un intero.

Altre assegnazioni saranno accettate ma potranno portare a problemi in fase di esecuzione

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

31

Slide a cura del dott. Claudio Musto

Variabili puntatore

- Come si assegna un valore alle variabili puntatore?

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

32

Variabili puntatore

- Come si assegna un valore alle variabili puntatore?
 - Il valore delle variabili puntatore è un indirizzo.
 - Abbiamo modo di conoscere direttamente gli indirizzi della macchina? NO
 - Si usa l'operatore indirizzo

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

33

Slide a cura del dott. Claudio Musto

Variabili puntatore

- Come si assegna un valore alle variabili puntatore?
 - Il valore delle variabili puntatore è un indirizzo.
 - Abbiamo modo di conoscere direttamente gli indirizzi della macchina? NO
 - Si usa l'operatore indirizzo

Cosa succede in memoria?

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

35

Slide a cura del dott. Claudio Musto

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t a: %X", a, aPtr);
14    printf("\n b: %.2f \t b: %X", b, bPtr);
15 }

```

Variabili puntatore

- Come si assegna un valore alle variabili puntatore?
 - Il valore delle variabili puntatore è un indirizzo.
 - Abbiamo modo di conoscere direttamente gli indirizzi della macchina? NO
 - Si usa l'operatore indirizzo

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

34

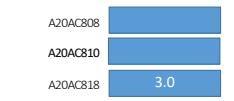
Slide a cura del dott. Claudio Musto

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t a: %X", a, aPtr);
14    printf("\n b: %.2f \t b: %X", b, bPtr);
15 }

```

Variabili puntatore



Situazione di partenza,
con le due variabili allocate

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

36

Slide a cura del dott. Claudio Musto

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t a: %X", a, aPtr);
14    printf("\n b: %.2f \t b: %X", b, bPtr);
15 }

```

Variabili puntatore

A20AC808 A20AC81C aPtr
A20AC810 A20AC818 bPtr
A20AC818 3.0 b
A20AC81C 5 a

Assegniamo ai puntatori i **due indirizzi delle variabili** (di tipo compatibile)

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = &a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t&a: %X", a, aPtr);
14    printf("\n b: %.2f \t&b: %X", b, bPtr);
15 }
```

Slide a cura del dott. Claudio Musto

Variabili puntatore

A20AC808 A20AC81C aPtr
A20AC810 A20AC818 bPtr
A20AC818 3.0 b
A20AC81C 5 a

Note: dopo l'assegnazione è possibile rappresentare graficamente i puntatori

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t&a: %X", a, aPtr);
14    printf("\n b: %.2f \t&b: %X", b, bPtr);
15 }
```

Slide a cura del dott. Claudio Musto

Variabili puntatore

A20AC808 A20AC81C aPtr
A20AC810 A20AC818 bPtr
A20AC818 3.0 b
A20AC81C 5 a

Stampiamo in output i quattro valori

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = &a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t&a: %X", a, aPtr);
14    printf("\n b: %.2f \t&b: %X", b, bPtr);
15 }
```

Slide a cura del dott. Claudio Musto

Variabili puntatore

A20AC808 A20AC818 bPtr
A20AC810 A20AC81C aPtr
A20AC818 3.0 b
A20AC81C 5 a

Stampiamo in output i quattro valori

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t&a: %X", a, aPtr);
14    printf("\n b: %.2f \t&b: %X", b, bPtr);
15 }
```

Slide a cura del dott. Claudio Musto

Operatore di indirezione

- Dato un puntatore, il linguaggio C mette a disposizione anche un operatore di **indirezione**
- A che serve?**
 - Serve a risalire al valore memorizzato nella cella di memoria dell'indirizzo puntato dalla variabile

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

41

Slide a cura del dott. Claudio Musto

Operatore di indirezione

- Dato un puntatore, il linguaggio C mette a disposizione anche un operatore di **indirezione**
- A che serve?**
 - Serve a risalire al valore memorizzato nella cella di memoria dell'indirizzo puntato dalla variabile

10/04/19

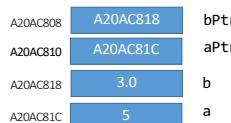
Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

42

Slide a cura del dott. Claudio Musto

Operatore di indirezione

- L'operatore di indirezione si esprime con l'asterisco accanto a una variabile di tipo puntatore (es. ***bPtr**)



Esempio: **bPtr** è un variabile di tipo puntatore. La variabile punta a un indirizzo. Attraverso l'operatore di indirezione posso risalire al valore memorizzato nella cella di memoria identificato dall'indirizzo.

Slide a cura del dott. Claudio Musto

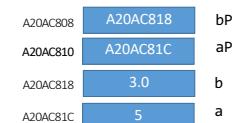
10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

43

Operatore di indirezione

- L'operatore di indirezione si esprime con l'asterisco accanto a una variabile di tipo puntatore (es. ***bPtr**)
- In questo caso il valore di ***bPtr** è uguale a 3.0



Esempio: **bPtr** è un variabile di tipo puntatore. La variabile punta a un indirizzo. Attraverso l'operatore di indirezione posso risalire al valore memorizzato nella cella di memoria identificato dall'indirizzo.

Slide a cura del dott. Claudio Musto

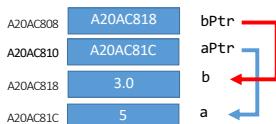
10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

44

Operatore di indirezione

- L'operatore di indirezione si esprime con l'asterisco accanto a una variabile di tipo puntatore (es. ***bPtr**)
- In questo caso il valore di ***bPtr** è uguale a 3.0
 - Il concetto di indirezione è più semplice se pensiamo alla rappresentazione grafica dei puntatori



10/04/19

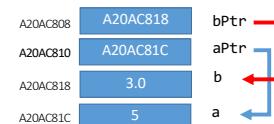
Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

45

Slide a cura del dott. Claudio Musto

Operatore di indirezione

- L'operatore di indirezione si esprime con l'asterisco accanto a una variabile di tipo puntatore (es. ***bPtr**)
- Attenzione a non fare confusione tra asterischi
 - Asterisco usato per dichiarare una variabile (**float* bPtr**)
 - Asterisco usato come operatore di indirezione sul puntatore (***bPtr**)



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

46

Slide a cura del dott. Claudio Musto

Recap

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t b: %f", a, *aPtr);
14    printf("\n b: %.2f \t &b: %x", b, bPtr);
15 }
  
```

a è un intero → il suo valore è 5
 b è un float → il suo valore è 3.0

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

47

Recap

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t b: %f", a, *aPtr);
14    printf("\n b: %.2f \t &b: %x", b, bPtr);
15 }
  
```

a è un intero → il suo valore è 5
 b è un float → il suo valore è 3.0

aPtr è un puntatore a un intero →
bPtr è un puntatore a un float →

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

48

Recap

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t&a: %X", a, aPtr);
14    printf("\n b: %.2f \t&b: %X", b, bPtr);
15 }

```

a è un intero → il suo valore è 5
b è un float → il suo valore è 3.0

aPtr è un puntatore a un intero →
il suo valore è l'indirizzo di **a**
bPtr è un puntatore a un float →
il suo valore è l'indirizzo di **b**

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

49

Slide a cura del dott. Cesare Musto

Recap

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t&a: %X", a, aPtr);
14    printf("\n b: %.2f \t&b: %X", b, bPtr);
15 }

```

a è un intero → il suo valore è 5
b è un float → il suo valore è 3.0

aPtr è un puntatore a un intero →
il suo valore è l'indirizzo di **a**
bPtr è un puntatore a un float →
il suo valore è l'indirizzo di **b**

***aPtr** è l'operatore di indirezione sul
puntatore **aPtr** →
***bPtr** è l'operatore di indirezione sul
puntatore **bPtr** →

Slide a cura del dott. Cesare Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

50

Recap

```

1 #include <stdio.h>
2 int main() {
3     int a = 5;
4     float b = 3.0;
5
6     int* aPtr; // puntatore a intero
7     float* bPtr; // puntatore a float
8
9     aPtr = a; // errore
10    aPtr = &a; // ok
11    bPtr = &b; // ok
12
13    printf("\n a: %d \t&a: %X", a, aPtr);
14    printf("\n b: %.2f \t&b: %X", b, bPtr);
15 }

```

a è un intero → il suo valore è 5
b è un float → il suo valore è 3.0

aPtr è un puntatore a un intero →
il suo valore è l'indirizzo di **a**
bPtr è un puntatore a un float →
il suo valore è l'indirizzo di **b**

***aPtr** è l'operatore di indirezione sul
puntatore **aPtr** → il suo valore è 5
***bPtr** è l'operatore di indirezione sul
puntatore **bPtr** → il suo valore è 3.0

Slide a cura del dott. Cesare Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

51

Recap

Quando usiamo l'operatore di indirezione
parlamo di «dereferenziare una variabile».
Dereferenziando il puntatore, in questo caso,
ritorniamo al valore della variabile di partenza.

```

8 #include <stdio.h>
9
10 int main() {
11     int a = 5;
12     float b = 3.0; // dichiaro le due variabili di input dell'esercizio
13
14     int* aPtr;
15     float* bPtr; // dichiaro due variabili di tipo puntatore
16
17     // aPtr = a --> ERRORE, devo assegnare un indirizzo
18     aPtr = &a;
19     bPtr = &b; // assegno ai puntatori due indirizzi di variabile "di tipo compatibile"
20
21     printf("Valore di a: %d \t\t Indirizzo di a: %X\n", a, aPtr);
22     printf("Valore di b: %.2f \t\t Indirizzo di b: %X\n", b, bPtr); // stampo i valori delle variabili
23
24     printf("Valore di aPtr: %X \t\t Indiruzione di aPtr: %d \n", aPtr, *aPtr);
25     printf("Valore di bPtr: %X \t\t Indiruzione di bPtr: %.2f \n", bPtr, *bPtr); // stampo i valori delle variabili
26
27 }
28

```

Slide a cura del dott. Cesare Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

52

Recap

```

8 #include <stdio.h>
9
10 int main() {
11     int a = 5;
12     float b = 3.0; // dichiaro le due variabili di input dell'esercizio
13
14     int* aPtr;
15     float* bPtr; // dichiaro due variabili di tipo puntatore
16
17     // aPtr = a --> ERRORE, devo assegnare un indirizzo
18     aPtr = &a;
19     bPtr = &b; // assegno ai puntatori due indirizzi di variabile "di tipo compatibile"
20
21     printf("Valore di a: %d\n", a);
22     printf("Valore di b: %f\n", b); // stampo i valori delle variabili
23
24     printf("Valore di aPtr: %X\n", aPtr);
25     printf("Valore di bPtr: %X\n", bPtr); // stampo i valori delle variabili
26
27 }
28 
```

L'indirizzo della variabile a (`&a`) corrisponde al puntatore a`aPtr`

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

53

Slide a cura del dott. Claudio Musto

Recap

```

8 #include <stdio.h>
9
10 int main() {
11     int a = 5;
12     float b = 3.0; // dichiaro le due variabili di input dell'esercizio
13
14     int* aPtr;
15     float* bPtr; // dichiaro due variabili di tipo puntatore
16
17     // aPtr = a --> ERRORE, devo assegnare un indirizzo
18     aPtr = &a;
19     bPtr = &b; // assegno ai puntatori due indirizzi di variabile "di tipo compatibile"
20
21     printf("Valore di a: %d\n", a);
22     printf("Valore di b: %f\n", b); // stampo i valori delle variabili
23
24     printf("Valore di aPtr: %X\n", aPtr);
25     printf("Valore di bPtr: %X\n", bPtr); // stampo i valori delle variabili
26
27 }
28 
```

L'indirizzo della variabile a (`&a`) corrisponde al puntatore a`aPtr`

Il valore dereferenziato di a`aPtr` corrisponde al valore della variabile a

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

54

Slide a cura del dott. Claudio Musto

Note importanti (1)

- Attenzione a non fare confusione tra asterischi
 - Asterisco usato per dichiarare una variabile (`float* bPtr`)
 - Asterisco usato come operatore di indirezione sul puntatore (`*bPtr`)
- In una dichiarazione, l'asterisco serve per **dichiarare una variabile** di tipo puntatore
 - Es.: `int *pi;`
- In una espressione, l'asterisco funge da **operatore di dereferenziazione**
 - Es.: `b = *pi;`

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

55

Slide a cura del dott. Claudio Musto

Note importanti (2)

- Gli operatori di dereferenziazione (`*`) e di indirizzo (`&`) sono uno l'inverso dell'altro. **Che significa?**

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

56

Slide a cura del dott. Claudio Musto

Note importanti (2)

- Gli operatori di dereferenziazione (*****) e di indirizzo (**&**) sono uno l'inverso dell'altro. **Che significa?**
- data la dichiarazione **int a**
 - ***&a** equivale a scrivere **a**
 - (Il valore memorizzato nell'indirizzo di memoria della variabile **a == a**)
- data la dichiarazione **int *pi;**
 - **&*pi** ha valore uguale a **pi**
 - (L'indirizzo di memoria del contenuto puntato dal puntatore **pi == pi**)

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

57

Slide a cura del dott. Claudio Musto

Note importanti (3)

- Anche le variabili puntatore devono essere **inizializzate**
- Di base il compilatore assegna un indirizzo random alle variabili puntatore
 - **Problema:** non sappiamo cosa è memorizzato in quelle locazioni di memoria!

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

58

Slide a cura del dott. Claudio Musto

Note importanti (3)

- Anche le variabili puntatore devono essere **inizializzate**
- Di base il compilatore assegna un indirizzo random alle variabili puntatore
 - **Problema:** non sappiamo cosa è memorizzato in quelle locazioni di memoria!
 - **Ulteriori operazioni fatte con i puntatori – senza inizializzare la variabile - rischiano di rendere corrotta la memoria e di creare problemi in esecuzione**

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

59

Slide a cura del dott. Claudio Musto

Note importanti (3)

- Anche le variabili puntatore devono essere **inizializzate**
- Di base il compilatore assegna un indirizzo random alle variabili puntatore
 - **Problema:** non sappiamo cosa è memorizzato in quelle locazioni di memoria!
 - **Ulteriori operazioni fatte con i puntatori – senza inizializzare la variabile - rischiano di rendere corrotta la memoria e di creare problemi in esecuzione**

Esempio

```
int a; int* pi; // puntatore non inizializzato
int b = 3; // variabile inizializzata
a = *pi;
*pi = 500;
```



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

60

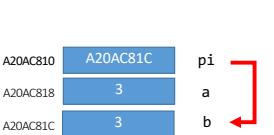
Slide a cura del dott. Claudio Musto

Note importanti (3)

- Anche le variabili puntatore devono essere **inizializzate**
- Di base il compilatore assegna un indirizzo random alle variabili puntatore
 - **Problema:** non sappiamo cosa è memorizzato in quelle locazioni di memoria!
 - **Ulteriori operazioni fatte con i puntatori – senza inizializzare la variabile - rischiano di rendere corrotta la memoria e di creare problemi in esecuzione**

• Esempio

```
int a; int* pi; // puntatore non inizializzato
int b = 3; // variabile inizializzata
a = *pi; // assegno ad a il valore puntato
*pi = 500;
```



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

61

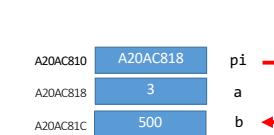
Slide a cura del dott. Claudio Musto

Note importanti (3)

- Anche le variabili puntatore devono essere **inizializzate**
- Di base il compilatore assegna un indirizzo random alle variabili puntatore
 - **Problema:** non sappiamo cosa è memorizzato in quelle locazioni di memoria!
 - **Ulteriori operazioni fatte con i puntatori – senza inizializzare la variabile - rischiano di rendere corrotta la memoria e di creare problemi in esecuzione**

• Esempio

```
int a; int* pi; // puntatore non inizializzato
int b = 3 // variabile inizializzata
a = *pi; // assegno ad a il valore puntato
*pi = 500 // senza volerlo ho corrotto il valore della variabile b!
```



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

62

Slide a cura del dott. Claudio Musto

Note importanti (3)

- Anche le variabili puntatore devono essere **inizializzate**
- Di base il compilatore assegna un indirizzo random alle variabili puntatore
 - **Problema:** non sappiamo cosa è memorizzato in quelle locazioni di memoria!
 - **Ulteriori operazioni fatte con i puntatori – senza inizializzare la variabile - rischiano di rendere corrotta la memoria e di creare problemi in esecuzione**
- Per **inizializzare le variabili puntatore** si utilizza **NULL**
 - `int x;`
 - `int *pi = NULL;`
 - `float *pf = NULL`

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

63

Slide a cura del dott. Claudio Musto



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

64

Slide a cura del dott. Claudio Musto

Passaggio dei Parametri

- La principale applicazione dei puntatori è legata al passaggio dei parametri
- In C i parametri vengono passati per valore
 - Il valore del parametro attuale viene copiato nel parametro formale (che è una variabile locale della funzione)
 - E' il metodo più sicuro per evitare modifiche accidentali ai parametri
 - Se all'interno di una funzione effettuiamo delle modifiche ai valori dei parametri, queste modifiche vengono perse!
- A volte questo non è sufficiente!

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

65

Slide a cura del dott. Claudio Musto

Passaggio dei Parametri - Esempio

```
void scambia(int a, int b) {
    int t; // variabile locale di appoggio
    t = a; // scambio dei valori
    a = b;
    b = t;
}

main() {
    int x = 33, y = 5;
    scambia(x, y);
    printf("x = %d, y = %d\n", x, y);
}
```

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

66

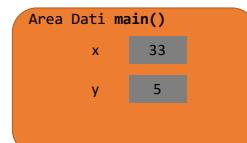
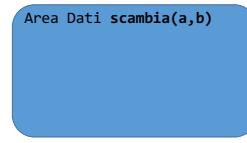
Slide a cura del dott. Claudio Musto

Passaggio dei Parametri - Esempio

```
void scambia(int a, int b) {
    int t; // variabile locale di appoggio
    t = a; // scambio dei valori
    a = b;
    b = t;
}

main() {
    int x = 33, y = 5;
    scambia(x, y);
    printf("x = %d, y = %d\n", x, y);
}
```

1. Inizializzazione



Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

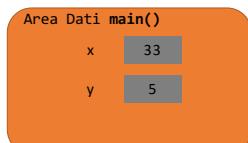
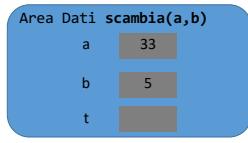
67

Passaggio dei Parametri - Esempio

```
void scambia(int a, int b) {
    int t; // variabile locale di appoggio
    t = a; // scambio dei valori
    a = b;
    b = t;
}

main() {
    int x = 33, y = 5;
    scambia(x, y);
    printf("x = %d, y = %d\n", x, y);
}
```

2. Chiamata della Funzione



Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

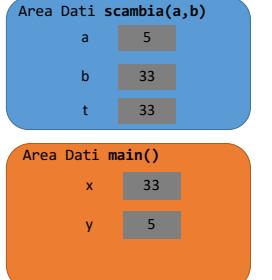
68

Passaggio dei Parametri - Esempio

```
void scambia(int a, int b) {
    int t; // variabile locale di appoggio
    t = a; // scambio dei valori
    a = b;
    b = t;
}

main() {
    int x = 33, y = 5;
    scambia(x, y);
    printf("x = %d, y = %d\n", x, y);
}
```

3.
Scambio dei Valori



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

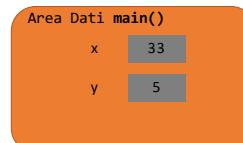
69

Passaggio dei Parametri - Esempio

```
void scambia(int a, int b) {
    int t; // variabile locale di appoggio
    t = a; // scambio dei valori
    a = b;
    b = t;
}

main() {
    int x = 33, y = 5;
    scambia(x, y);
    printf("x = %d, y = %d\n", x, y);
}
```

4.
Uscita dalla Funzione



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

70

Passaggio dei Parametri - Esempio

```
void scambia(int a, int b) {
    int t; // variabile locale di appoggio
    t = a; // scambio dei valori
    a = b;
    b = t;
}

main() {
    int x = 33, y = 5;
    scambia(x, y);
    printf("x = %d, y = %d\n", x, y);
}
```

4.
Uscita dalla Funzione



10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

71

Passaggio dei Parametri - Esempio

```
void scambia(int* a, int* b) {
    int t; // variabile locale di appoggio
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;
}

main() {
    int x = 33, y = 5;
    scambia(&x, &y);
    printf("x = %d, y = %d\n", x, y);
}
```

Attraverso l'utilizzo dei puntatori possiamo simulare il passaggio per riferimento, che risolve questi problemi.

Invece di passare il valore delle variabili, ne passiamo l'indirizzo! In questo modo, le modifiche vengono ereditate dalle variabili originali

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

Slide 7 di 10 di Claudio Musto

72

Passaggio dei Parametri - Esempio

```
void scambia(int* a, int* b)
    int t; // variabile locale di appoggio
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;

main() {
    int x = 33, y = 5;
    scambia(&x, &y);
    printf("x = %d, y = %d\n", x, y);
}
```

Attraverso l'utilizzo dei **puntatori** possiamo simulare il **passaggio per riferimento**, che risolve questi problemi.

Invece di passare il **valore** delle variabili, ne **passiamo l'indirizzo**! In questo modo, le modifiche vengono ereditate dalle variabili originali

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

73

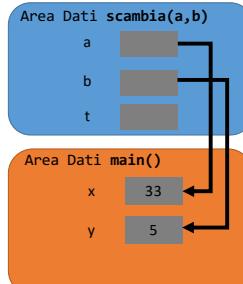
Slide a cura del dott. Claudio Musto

Passaggio dei Parametri - Esempio

```
void scambia(int* a, int* b) {
    int t; // variabile locale di appoggio
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;

main() {
    int x = 33, y = 5;
    scambia(&x, &y);
    printf("x = %d, y = %d\n", x, y);
}
```

2. Chiamata della Funzione



Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

75

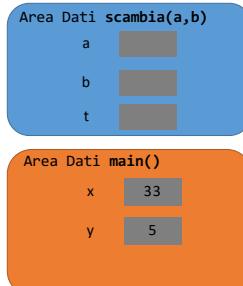
Slide a cura del dott. Claudio Musto

Passaggio dei Parametri - Esempio

```
void scambia(int* a, int* b) {
    int t; // variabile locale di appoggio
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;

main() {
    int x = 33, y = 5;
    scambia(&x, &y);
    printf("x = %d, y = %d\n", x, y);
}
```

1. Inizializzazione



Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

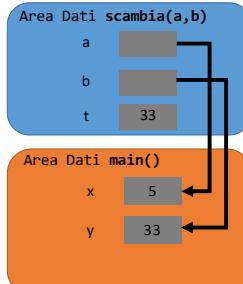
74

Passaggio dei Parametri - Esempio

```
void scambia(int* a, int* b) {
    int t; // variabile locale di appoggio
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;

main() {
    int x = 33, y = 5;
    scambia(&x, &y);
    printf("x = %d, y = %d\n", x, y);
}
```

3. Scambio dei valori



Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

76

Passaggio dei Parametri - Esempio

```
void scambia(int* a, int* b) {
    int t; // variabile locale di appoggio
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;
}

main() {
    int x = 33, y = 5;
    scambia(&x, &y);
    printf("x = %d, y = %d\n", x, y);
}
```

4.
Uscita dalla
funzione

Area Dati main()	
x	5
y	33



Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

77

Passaggio dei Parametri - Recap

- La principale applicazione dei puntatori è legata al passaggio dei parametri
- Attraverso l'utilizzo dei puntatori **possiamo simulare** il passaggio per riferimento, **passando alla funzione l'indirizzo delle variabili** invece che i loro valori

• Quando serve?

- Serve **NECESSARIAMENTE** quando la funzione deve modificare i valori dei **parametri** (es. scambio di valori). Esistono invece altre situazioni in cui è consigliabile, ma non obbligatorio.

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

78

Slide a cura del dott. Claudio Musto

Utilizzo dei puntatori

- Una ulteriore applicazione dei puntatori è legata alla possibilità di far restituire alle funzioni più di un valore
- Le funzioni, di base, possono restituire solo un valore!**

Slide a cura del dott. Claudio Musto

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

79

Utilizzo dei puntatori

- Una ulteriore applicazione dei puntatori è legata alla possibilità di far restituire alle funzioni più di un valore
- Le funzioni, di base, possono restituire solo un valore!**

• Esempio

```
int scambio-somma(int* a, int* b) {
    int t; int somma;
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;

    somma = *a + *b;
    return somma;
}
```

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

80

Slide a cura del dott. Claudio Musto

Utilizzo dei puntatori

- Una ulteriore applicazione dei puntatori è legata alla possibilità di far restituire alle funzioni più di un valore
- Le funzioni, di base, possono restituire solo un valore!**

Esempio

```
int scambio-somma(int* a, int* b) {
    int t; int somma;
    t = *a; // scambio dei valori
    *a = *b;
    *b = t;

    somma = *a + *b;
    return somma;
}
```

Formalmente, la funzione restituisce solo un valore intero, ma in realtà restituisce tre valori perché **scambia anche i valori di a e b.**

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

81

Slide a cura del dott. Claudio Mazzoli

Esercizio 8.1

- Scrivere un programma che generi random il tempo sul giro, in millisecondi, di cinque diverse macchine di Formula 1. Il tempo deve essere compreso tra 1 minuto e 20 e 1 minuto e 30. I valori devono essere memorizzati in un vettore.**
- Scrivere una funzione che converta il tempo in millisecondi in minuti, secondi e decimi di secondo, e li mostri in output
 - Suggerimento: utilizzare i puntatori per restituire più di un valore
 - Esempio:** 82738 ms. = 1' 22" 738
- Scrivere una funzione che ordini i tempi dal più rapido al più lento (opzionale)**
 - Suggerimento: utilizzare lo scambio e il passaggio dei parametri per riferimento

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

82

Slide a cura del dott. Claudio Mazzoli

DOMANDE?



imgflip.com

10/04/19

Veronica Rossano - Puntatori (Parte 1) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

90

Slide a cura del dott. Claudio Mazzoli