



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

# Laboratorio di Informatica

## Puntatori

docente: Veronica Bossano

(Parte 2)

[veronica.rossano@uniiba.it](mailto:veronica.rossano@uniiba.it)

veronica.rossano@uniba.it

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

- ## Aritmetica dei Puntatori

- I puntatori sono inoltre particolarmente utili per manipolare e accedere agli elementi di un array, attraverso un meccanismo che prende il nome di **aritmetica dei puntatori**
  - **Concetti che già conosciamo**

**Il nome dell'array è un puntatore al primo elemento dell'array**

29/04/19

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

## Aritmetica dei Puntatori

- I puntatori sono inoltre particolarmente utili per manipolare e accedere agli elementi di un array, attraverso un meccanismo che prende il nome di **aritmetica dei puntatori**
  - **Concetti che già conosciamo**

**Il nome dell'array è un puntatore al primo elemento dell'array.**

```
int array[10] // dichiaro un array di interi  
int* p_array // dichiaro un puntatore a un intero
```

```
p_array = &array[0];  
p_array = array; // scrittura equivalente
```

29/04/19

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

## Aritmetica dei Puntatori

```
1 #include <stdio.h>
2
3 int main() {
4     int array[10]; // dichiaro un array di interi
5     int* p_array; // dichiaro un puntatore a un intero
6
7     p_array = &array[0];
8
9     printf("Nome dell'array: \t\t% s \n", array);
10    printf("Indirizzo del primo elemento: \t% x \n", &array[0]);
11    printf("Puntatore all'array: \t\t% x \n", p_array);
12 }
13
```

## Cosa stampa?

29/04/19

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

## Aritmetica dei Puntatori

```
1 #include <stdio.h>
2
3 + int main() {
4     int array[10]; // dichiaro un array di interi
5     int* p_array; // dichiaro un puntatore a un intero
6
7     p_array = &array[0];
8
9     printf("Nome dell'array: %s\n", array);
10    printf("Indirizzo del primo elemento: %p\n", &array[0]);
11    printf("Puntatore all'array: %p\n", p_array);
12 }
```

## Cosa stampa?

29/04/19

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

5

## Aritmetica dei Puntatori

```
1 #include <stdio.h>
2
3 + int main() {
4     int array[10]; // dichiaro un array di interi
5     int* parray; // dichiaro un puntatore a un intero
6
7     p_array = &array[0];
8
9     printf("Nome dell'array: %s\n", array);
10    printf("Indirizzo del primo elemento: %s\n", &array[0]);
11    printf("Puntatore all'array: %s\n", p_array);
12 }
```

## Cosa stampa?

```
gcc version 4.6.3
>
Nome dell'array: A9FBE990
Indirizzo del primo elemento: A9FBE990
Puntatore all'array: A9FBE990
```

**Le tre scritture sono equivalenti.** Il nome dell'array è un indirizzo di memoria, che corrisponde all'indirizzo di memoria del primo elemento dell'array, che a sua volta può tranquillamente essere assegnato a un puntatore

29/04/19

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

7

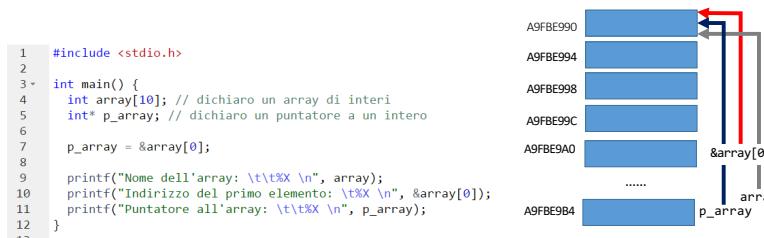
## Aritmetica dei Puntatori

Cosa stampa?

```
gcc version 4.6.3
Nome dell'array: A9FBE900
Indirizzo del primo elemento: A9FBE900
Puntatore all'array: A9FBE900
```

29/04/19

Veronica Rossano - Puntatori (Parte 2) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/20



**Le tre scritture sono equivalenti.** Il nome dell'array è un indirizzo di memoria, che corrisponde all'indirizzo di memoria del primo elemento dell'array, che a sua volta può tranquillamente essere assegnato a un puntatore

29/04/19

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

## Aritmetica dei Puntatori

- Array e puntatori sono in stretta relazione
- Attraverso l'aritmetica dei puntatori possiamo accedere e manipolare gli elementi di un array in modo «alternativo»

29/04/19

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

9

## Aritmetica dei Puntatori

- Array e puntatori sono in stretta relazione
- Attraverso l'aritmetica dei puntatori possiamo accedere e manipolare gli elementi di un array in modo «alternativo»
- Come accediamo normalmente agli elementi di un array?



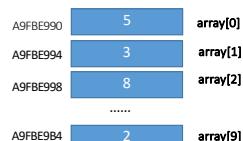
29/04/19

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

10

## Aritmetica dei Puntatori

- Array e puntatori sono in stretta relazione
- Attraverso l'aritmetica dei puntatori possiamo accedere e manipolare gli elementi di un array in modo «alternativo»
- Come accediamo normalmente agli elementi di un array?
  - Con la notazione indice (es. `array[i]` → i+1 esimo elemento del vettore)



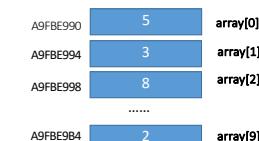
29/04/19

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

11

## Aritmetica dei Puntatori

- Array e puntatori sono in stretta relazione
- Attraverso l'aritmetica dei puntatori possiamo accedere e manipolare gli elementi di un array in modo «alternativo»
- Come accediamo normalmente agli elementi di un array?
  - Con la notazione indice (es. `array[i]` → i+1 esimo elemento del vettore)



Attraverso l'aritmetica  
dei puntatori possiamo  
utilizzare i puntatori in  
espressioni aritmetiche.

29/04/19

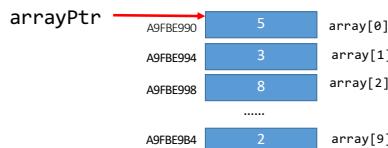
Veronica Rossano - Puntatori (Parte 2) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

12

## Aritmetica dei Puntatori - Esempio

- Attraverso l'aritmetica dei puntatori **possiamo utilizzare i puntatori in espressioni aritmetiche.**

• `int* arrayPtr = &array[0]; // punta al primo elemento`



29/04/19

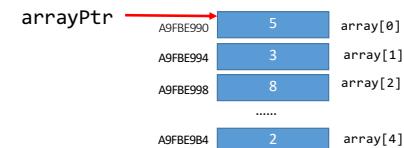
Veronica Rossano - Puntatori (Parte 2) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

13

## Aritmetica dei Puntatori - Esempio

- Attraverso l'aritmetica dei puntatori **possiamo utilizzare i puntatori in espressioni aritmetiche.**

• `int* arrayPtr = &array[0]; // punta al primo elemento`  
 • `arrayPtr= arrayPtr+4; // che cosa succede?`



29/04/19

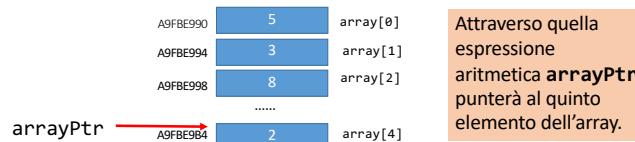
Veronica Rossano - Puntatori (Parte 2) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

14

## Aritmetica dei Puntatori - Esempio

- Attraverso l'aritmetica dei puntatori **possiamo utilizzare i puntatori in espressioni aritmetiche.**

• `int* arrayPtr = &array[0]; // punta al primo elemento`  
 • `arrayPtr= arrayPtr+4; // che cosa succede?`



29/04/19

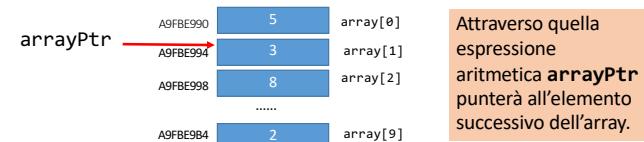
Veronica Rossano - Puntatori (Parte 2) Laboratorio di Informatica (INF, Track B) – Università degli Studi di Bari – A.A. 2018/2019

15

## Aritmetica dei Puntatori - Esempio

- Attraverso l'aritmetica dei puntatori **possiamo utilizzare i puntatori in espressioni aritmetiche.**

• `int* arrayPtr = &array[0]; // punta al primo elemento`  
 • `arrayPtr= arrayPtr+1; // che cosa succede?`



29/04/19

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

16

## Aritmetica dei puntatori

- E se volessimo usare l'aritmetica dei puntatori per accedere al valore dei singoli elementi di un array?

29/04/19

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

21

## Aritmetica dei puntatori

- E se volessimo usare l'aritmetica dei puntatori per accedere al valore dei singoli elementi di un array?
- Bisogna usare l'operatore di indirezione, per accedere al contenuto dei puntatori**

29/04/19

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

22

## Aritmetica dei puntatori

- E se volessimo usare l'aritmetica dei puntatori per accedere al valore **dei singoli elementi di un array?**
- Bisogna usare l'operatore di indirezione, per accedere al contenuto dei puntatori**

Dato un array a di elementi di tipo t (t a[])

- a è il puntatore al primo elemento dell'array, pertanto:**
  - a[0]** accede al primo elemento dell'array
  - \*a** accede al primo elemento dell'array

29/04/19

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

23

## Aritmetica dei puntatori

- E se volessimo usare l'aritmetica dei puntatori per accedere al valore **dei singoli elementi di un array?**
- Bisogna usare l'operatore di indirezione, per accedere al contenuto dei puntatori**

Dato un array a di elementi di tipo t (t a[])

- a è il puntatore al primo elemento dell'array, pertanto:**
  - a[0]** accede al primo elemento dell'array
  - \*a** accede al primo elemento dell'array
- a+i punta all'(i+1)-esimo elemento dell'array, pertanto**
  - a[i]** accede allo (i+1)-esimo elemento dell'array
  - \*(a+i)** accede allo (i+1)-esimo elemento dell'array

29/04/19

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

24

## Aritmetica dei puntatori – Recap Generale

A724DC0	1	array[0]
A724DC4	2	array[1]
A724DC8	3	array[2]
A724DCC	4	array[3]
A724DD0	5	array[4]

```
int array[5] = {1, 2, 3, 4, 5}
```

29/04/19

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

25

## Aritmetica dei puntatori – Recap Generale

A724DC0	1	array[0] == array_ptr
A724DC4	2	array[1] == array_ptr + 1
A724DC8	3	array[2] == array_ptr + 2
A724DCC	4	array[3] == array_ptr + 3
A724DD0	5	array[4] == array_ptr + 4

```
int array[5] = {1, 2, 3, 4, 5}
int* array_ptr = &array[0]
```

29/04/19

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

26

## Aritmetica dei puntatori – Recap Generale

A724DC0	1	array[0] == array_ptr
A724DC4	2	array[1] == array_ptr + 1
A724DC8	3	array[2] == array_ptr + 2
A724DCC	4	array[3] == array_ptr + 3
A724DD0	5	array[4] == array_ptr + 4

La prima si chiama «notazione indice»  
La seconda si chiama «notazione puntatore + offset»

```
int array[5] = {1, 2, 3, 4, 5}
int* array_ptr = &array[0]

array[3] == *(array_ptr+3) // le notazioni sono equivalenti!
```

29/04/19

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

27

## Utilizzo dei puntatori - Passaggio degli Array

- Concetti che già conosciamo

Quando sono utilizzati come parametri nelle funzioni, gli array vengono passati automaticamente per riferimento.

Ora riusciamo a capire meglio il perché

28

## Utilizzo dei puntatori - Passaggio degli Array

- **Concetti che già conosciamo**

Quando sono utilizzati come parametri nelle funzioni, gli array vengono passati automaticamente per riferimento.

**Ora riusciamo a capire meglio il perché**

Perché il nome dell'array è un puntatore al primo elemento, quindi passandolo ad una funzione stiamo implicitamente passando l'indirizzo (dunque realizziamo senza saperlo un passaggio dei parametri per riferimento)

29/04/19

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

29

## Utilizzo dei puntatori - Passaggio degli Array

- **Concetti che già conosciamo**

Quando sono utilizzati come parametri nelle funzioni, gli array vengono passati automaticamente per riferimento.

**Ora riusciamo a capire meglio il perché**

Perché il nome dell'array è un puntatore al primo elemento, quindi passandolo ad una funzione stiamo implicitamente passando l'indirizzo (dunque realizziamo senza saperlo un passaggio dei parametri per riferimento)

**Esempio**

- Dato l'array `int vector[3]` e data una funzione `double f(int v[], n){...}`
- Quando si invoca `f(vector, 3)`, poiché `vector` è un puntatore al primo elemento dell'array, stiamo passando un puntatore e non l'intero array

29/04/19

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

30

## Utilizzo dei puntatori – Passaggio degli Array

```
int sum(int v[], int n)
```

- indica espressamente che `v` è un array
- **Più indicato usare la sintassi degli array**

```
int sum(int* v, int n)
```

- Non è chiaro che `v` è un array
- Più indicato se si usa l'aritmetica dei puntatori

L'array può essere passato alla funzione in entrambi i modi. La scelta è personale. La prima è probabilmente più leggibile e più comprensibile. La seconda è più utile se si utilizza l'aritmetica dei puntatori dentro la funzione.

29/04/19

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

31

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
  - Solo che possono contenere degli indirizzi

29/04/19

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

32

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
    - Solo che possono contenere degli indirizzi
  - I puntatori possono anche essere inseriti in un array o in una struct
- Esempio**

```
int x = 33; // dichiaro i valori int
y = 5;
```



29/04/19

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

33

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
    - Solo che possono contenere degli indirizzi
  - I puntatori possono anche essere inseriti in un array o in una struct
- Esempio**

```
int x = 33; // dichiaro i valori int
```

```
y = 5;
```

```
// array di puntatori ad interi
```

```
int* array[2] = // come lo inizializzo?
```



29/04/19

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

34

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
    - Solo che possono contenere degli indirizzi
  - I puntatori possono anche essere inseriti in un array o in una struct
- Esempio**

```
int x = 33; // dichiaro i valori int
y = 5;
// array di puntatori ad interi
int* array[2] = {&x, &y};
```



29/04/19

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

35

## Array di Puntatori

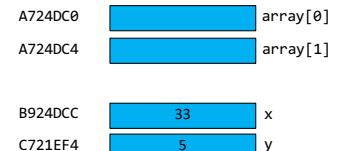
- I puntatori sono delle variabili a tutti gli effetti
    - Solo che possono contenere degli indirizzi
  - I puntatori possono anche essere inseriti in un array o in una struct
- Esempio**

```
int x = 33; // dichiaro i valori int
```

```
y = 5;
```

```
// array di puntatori ad interi
```

```
int* array[2] = {&x, &y};
```



29/04/19

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

36

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
  - Solo che possono contenere degli indirizzi
- I puntatori possono anche essere inseriti in un array o in una struct
- Esempio**

```
int x = 33; // dichiaro i valori int
y = 5;
// array di puntatori ad interi
int* array[2] = {&x, &y};
```



29/04/19

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

37

## Problema

- Riscriviamo la funzione **scambia()** facendo in modo che prenda in input un array di puntatori, invece che due puntatori

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

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

29/04/19

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

38

## Problema

- Riscriviamo la funzione **scambia()** facendo in modo che prenda in input un array di puntatori, invece che due puntatori

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

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

29/04/19

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

39

## Problema

- Riscriviamo la funzione **scambia()** facendo in modo che prenda in input un array di puntatori, invece che due puntatori

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

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

29/04/19

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

40

## Problema

- Riscriviamo la funzione **scambia()** facendo in modo che prenda in input un array di puntatori, invece che due puntatori

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

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

29/04/19

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

41

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a[0]); // scambio dei valori
    *(a[0]) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y}
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```

Dichiaro il vettore di puntatori



## Problema

- Riscriviamo la funzione **scambia()** facendo in modo che prenda in input un array di puntatori, invece che due puntatori

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

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

29/04/19

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

42

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a[0]); // scambio dei valori
    *(a[0]) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y}
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```

Cambiiamo l'input della funzione



## Problema

- Riscriviamo la funzione **scambia()** facendo in modo che prenda in input un array di puntatori, invece che due puntatori

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

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

29/04/19

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

43

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a[0]); // scambio dei valori
    *(a[0]) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y}
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```

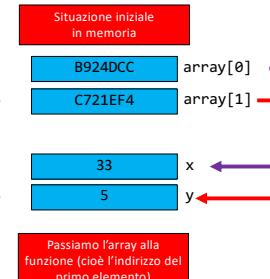
Modifichiamo la procedura di scambio

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a[0]); // scambio dei valori
    *(a[0]) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y}
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```



29/04/19

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

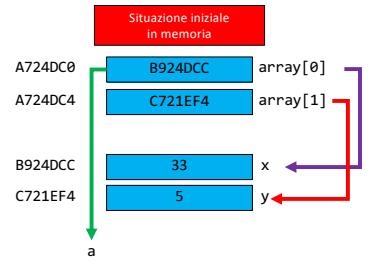
44

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a); // scambio dei valori
    *(a) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y};
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```



29/04/19

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

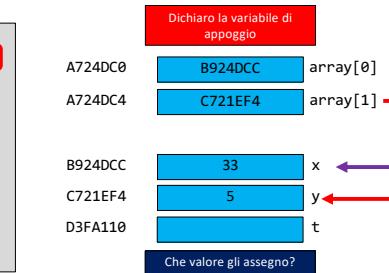
45

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a); // scambio dei valori
    *(a) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y};
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```



29/04/19

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

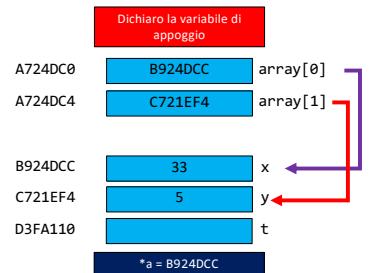
46

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a); // scambio dei valori
    *(a) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y};
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```



29/04/19

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

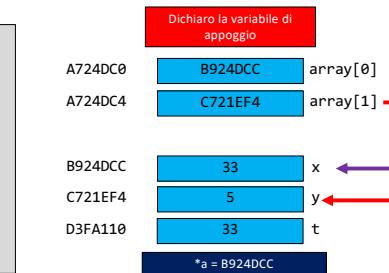
47

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a); // scambio dei valori
    *(a) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y};
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```



29/04/19

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

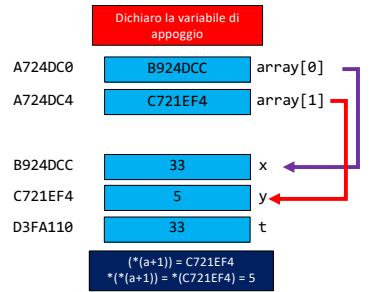
48

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a); // scambio dei valori
    *(a) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y};
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```



29/04/19

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

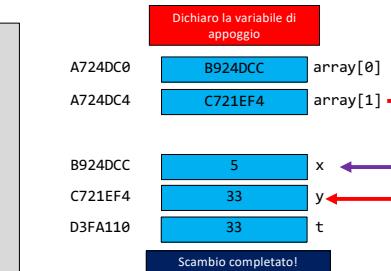
49

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a); // scambio dei valori
    *(a) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y};
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```



29/04/19

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

50

## Array di Puntatori - Esempio

- Studiamo meglio la procedura di scambio

```
void scambia(int* a[]) {
    int t; // variabile locale di appoggio
    t = *(a); // scambio dei valori
    *(a) = *(a+1);
    *(a+1) = t;
}

int main() {
    int x = 33, y = 5;
    int* array[2] = {&x, &y};
    scambia(array);
    printf("x = %d, y = %d\n", x, y);
}
```

Quando si utilizzano i vettori di puntatori è necessario utilizzare la **notazione con il doppio operatore di indirezione**.  
Il primo operatore di indirezione serve a risalire all'indirizzo di memoria del primo elemento.  
Il secondo operatore di indirezione serve a risalire al suo valore, che è quello che ci serve.

29/04/19

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

51

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
  - Solo che possono contenere degli indirizzi
- I puntatori possono anche essere inseriti in un array o in una struct

29/04/19

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

52

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
  - Solo che possono contenere degli indirizzi
- I puntatori possono anche essere inseriti in un array o in una struct
  - Applicazioni:** un array di puntatori ci serve ad esempio se abbiamo bisogno di un array di stringhe
  - Una struct con dentro un puntatore (o più puntatori) serve se dobbiamo implementare **delle strutture dati** (es. liste, pile)

29/04/19

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

53

## Array di Puntatori

- I puntatori sono delle variabili a tutti gli effetti
  - Solo che possono contenere degli indirizzi
- I puntatori possono anche essere inseriti in un array o in una struct
  - Applicazioni:** un array di puntatori ci serve ad esempio se abbiamo bisogno di un array di stringhe ← ci interessa ☺
  - Una struct con dentro un puntatore (o più puntatori) serve se dobbiamo implementare **delle strutture dati** (es. liste, pile) ← non ci interessa

29/04/19

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

54

## Array di Puntatori

- Perché un array di stringhe è un array di puntatori?

29/04/19

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

55

## Array di Puntatori

- Perché un array di stringhe è un array di puntatori?
  - Perché una stringa è un array di caratteri

29/04/19

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

56

## Array di Puntatori

- Perché un array di stringhe è un array di puntatori?
  - Perché una stringa è un array di caratteri
  - Un array di caratteri è un puntatore al primo caratteri

29/04/19

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

57

## Array di Puntatori

- Perché un array di stringhe è un array di puntatori?
  - Perché una stringa è un array di caratteri
  - Un array di caratteri è un puntatore al primo caratteri
  - Una stringa è un puntatore al primo carattere

29/04/19

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

58

## Array di Puntatori

- Perché un array di stringhe è un array di puntatori?
  - Perché una stringa è un array di caratteri
  - Un array è un puntatore al primo elemento (in questo caso al primo carattere)
  - Una stringa è un puntatore al primo carattere
  - Un array di stringhe è dunque un array di puntatori al primo carattere della stringa

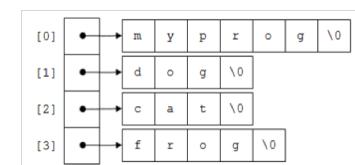
29/04/19

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

59

## Array di Puntatori

- Perché un array di stringhe è un array di puntatori?
  - Perché una stringa è un array di caratteri
  - Un array è un puntatore al primo elemento (in questo caso al primo carattere)
  - Una stringa è un puntatore al primo carattere
  - Un array di stringhe è dunque un array di puntatori al primo carattere della stringa



Vediamo come codificarlo!

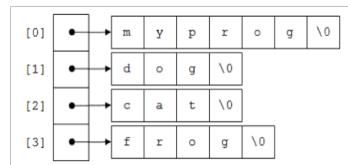
29/04/19

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

60

## Array di Puntatori

```
char* array[4] = {"myprog", "dog", "cat" , "frog"};
```



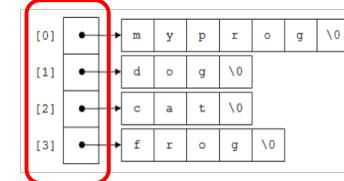
29/04/19

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

61

## Array di Puntatori

```
char* array[4] = {"myprog", "dog", "cat" , "frog"};
```



Istanziamo un  
array di 4 puntatori

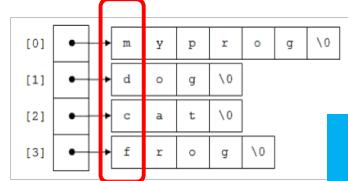
29/04/19

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

62

## Array di Puntatori

```
char* array[4] = {"myprog", "dog", "cat" , "frog"};
```



Ogni elemento è un  
puntatore a un char  
(quindi una stringa)

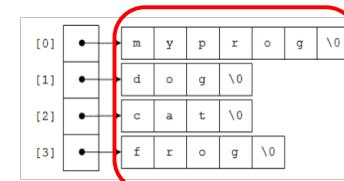
29/04/19

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

63

## Array di Puntatori

```
char* array[4] = {"myprog", "dog", "cat" , "frog"};
```



Importante: gli elementi  
possono anche avere  
lunghezza diversa

29/04/19

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

64

## Esercizio 9.1

- Scrivere una FUNZIONE che, dati in input due array di stringhe composti da nome e cognome, inverta il nome dei due individui e lo stampi in output. La stampa deve avvenire nel main!

### Esempio:

- Input:** Mario Rossi , Luigi Bianchi
- Output:** Luigi Rossi, Mario Bianchi

### Nota

Non è necessario acquisire interattivamente i dati. I vettori possono anche essere inizializzati nel codice

29/04/19

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

65

## Esercizio 9.1 - Soluzione

```

1 #include <stdio.h>
2
3 // equivalente a: scambia(char* a[], char* b[])
4 void scambia(char** a, char** b) {
5     char* s;
6     s = *a;
7     *a = *b;
8     *b = s;
9 }
```

Scambio i valori puntati, seguendo sempre la stessa procedura.  
Il puntatore al carattere non è altro che la stringa.

29/04/19

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

67

## Esercizio 9.1 - Soluzione

```

10
11 int main() {
12     char* persona_A[2] = {"Mario", "Rossi"};
13     char* persona_B[2] = {"Luigi", "Bianchi"};
14
15     // stampa, scambia i valori e stampa di nuovo
16     printf("\nbefore\n---\n%*s \t %*s \n%*s \t %*s",
17           persona_A[0], persona_A[1], persona_B[0], persona_B[1]);
18
19     scambia(persona_A, persona_B);
20
21     printf("\n\nAfter\n---\n%*s \t %*s \n%*s \t %*s",
22           persona_A[0], persona_A[1], persona_B[0], persona_B[1]);
23 }
```

Dichiaro i due vettori (**riga 11 e 12**), stampo i valori prima (**riga 16-17**), invoco la procedura di scambio (**riga 19**) e stampo i valori invertiti (**riga 21-22**)

29/04/19

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

66

## Esercizio 9.1 - Soluzione

```

1 #include <stdio.h>
2
3 // equivalente a: scambia(char* a[], char* b[])
4 void scambia(char** a, char** b) {
5     char* s;
6     s = *a;
7     *a = *b;
8     *b = s;
9 }
```

Before
---
Mario Rossi
Luigi Bianchi
After
---
Luigi Rossi
Mario Bianchi

Scambio i valori puntati, seguendo sempre la stessa procedura.  
Il puntatore al carattere non è altro che la stringa.

E se avessi voluto invertire i cognomi?

29/04/19

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

68

## Esercizio 9.1 - Soluzione

```

3 // equivalente a: scambia(char* a[], char* b[])
4 void scambia(char** a, char** b) {
5     char* s;
6     s = *(a+1);
7     *(a+1) = *(b+1);
8     *(b+1) = s;
9 }
10

```

Before		
-----	Mario	Rossi
	Luigi	Bianchi
After		
-----	Mario	Bianchi
	Luigi	Rossi

Scambio i valori puntati, seguendo sempre la stessa procedura.

Il puntatore al carattere non è altro che la stringa.

**E se avessi voluto invertire i cognomi?**

E' sufficiente puntare all'elemento successivo.

29/04/19

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

69

## Restituire un Puntatore

- Nuovamente

- I puntatori sono variabili a tutti gli effetti.
- Possono essere dichiarate e possono essere utilizzate in array e struct
- Possono ovviamente anche essere restituiti dalle funzioni

29/04/19

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

70

## Restituire un Puntatore

- Nuovamente

- I puntatori sono variabili a tutti gli effetti.
- Possono essere dichiarate e possono essere utilizzate in array e struct
- Possono ovviamente anche essere restituiti dalle funzioni

- Per far restituire un puntatore ad una funzione bisogna però seguire **degli accorgimenti particolari**

29/04/19

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

72

## Restituire un Puntatore

- Nuovamente

- I puntatori sono variabili a tutti gli effetti.
- Possono essere dichiarate e possono essere utilizzate in array e struct
- Possono ovviamente anche essere restituiti dalle funzioni

- Per far restituire un puntatore ad una funzione bisogna però seguire **degli accorgimenti particolari**

- Motivo:** allocazione della memoria
- La memoria in C viene gestita staticamente.** Ciò significa che bisogna sapere a priori quali variabili saranno utilizzate e quando saranno grandi.

29/04/19

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

73

## Restituire un Puntatore

- Nuovamente
  - I puntatori sono variabili a tutti gli effetti.
  - Possono essere dichiarate e possono essere utilizzate in array e struct
  - Possono ovviamente anche essere restituiti dalle funzioni
- Per far restituire un puntatore ad una funzione bisogna però seguire **degli accorgimenti particolari**
  - Motivo: allocazione della memoria
  - La memoria in C viene gestita staticamente. Ciò significa che bisogna sapere a priori quali variabili saranno utilizzate e quando saranno grandi.
    - Questo è il motivo per cui le variabili devono tutte essere dichiarate prima dell'esecuzione e soprattutto bisogna assegnare un tipo alle variabili!

29/04/19

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

74

## Restituire un Puntatore

- Nuovamente
  - I puntatori sono variabili a tutti gli effetti.
  - Possono essere dichiarate e possono essere utilizzate in array e struct
  - Possono ovviamente anche essere restituiti dalle funzioni
- Per far restituire un puntatore ad una funzione bisogna però seguire **degli accorgimenti particolari**
  - Motivo: allocazione della memoria
  - La memoria in C viene gestita staticamente. Ciò significa che bisogna sapere a priori quali variabili saranno utilizzate e quando saranno grandi.
  - Nel caso dei puntatori può essere un problema.
    - Perché?

29/04/19

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

75

## Restituire un Puntatore

- Nel caso dei puntatori può essere un problema.
  - Immaginiamo di dover scrivere una funzione che restituisce una stringa.
  - Possiamo sapere a priori quanto sarà lunga una stringa? No!

29/04/19

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

76

## Restituire un Puntatore

- Nel caso dei puntatori può essere un problema.
  - Immaginiamo di dover scrivere una funzione che restituisce una stringa.
  - Possiamo sapere a priori quanto sarà lunga una stringa? No!
  - L'allocazione statica della memoria non è sufficiente
- Soluzione: Allocazione Dinamica della memoria
  - (La studieremo meglio più avanti)
  - Funzione `malloc( )` e `calloc( )`
  - Serve a «riservare» una quantità di memoria al nostro programma, non definita a priori
- Esempio: scrivere una funzione che restituisce una Stringa

29/04/19

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

77

## Restituire un Puntatore

- **Esempio:** scrivere una funzione che restituisce una Stringa

```
char* generatePassword (char* nome, char* cognome) {
    //alloco un numero sufficiente di caratteri
    char* s = (char*) calloc(10, sizeof(char)); ←
    ...
    // restituisco la nuova stringa
    return s;
}
```

29/04/19

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

78

## Restituire un Puntatore

- **Esempio:** scrivere una funzione che restituisce una Stringa

```
char* generatePassword (char* nome, char* cognome) {
    //alloco un numero sufficiente di caratteri
    char* s = (char*) calloc(10, sizeof(char)); →
    ...
    // restituisco la nuova stringa
    return s;
}
```

Chiede di allocare dinamicamente la memoria per 10 caratteri

29/04/19

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

79

## Restituire un Puntatore

- **Esempio:** scrivere una funzione che restituisce una Stringa

```
char* generatePassword (char* nome, char* cognome) {
    //alloco un numero sufficiente di caratteri
    char* s = (char*) calloc(10, sizeof(char)); →
    ...
    // restituisco la nuova stringa
    return s;
}
```

Chiede di allocare dinamicamente la memoria per 10 caratteri

Test: provate a eseguire un programma senza l'istruzione `calloc`. Cosa succede?

29/04/19

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

80

## Esercizio 9.2

- Scrivere una funzione `char* generaPassword (char* nome, char* cognome)` che generi random una password per l'utente
- La funzione prende in input nome e cognome (max. 20 caratteri)
- La funzione restituisce una stringa con i primi tre caratteri del nome, i primi tre caratteri del cognome e due simboli a caso
- Inserire la funzione in un `main()` che chieda interattivamente all'utente di inserire nome e cognome e stampi la password generata
- Utilizzare `calloc()` per allocare la memoria.

29/04/19

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

82

## Esercizio 9.2 - Soluzione

```
#include <stdio.h>
#include <stdlib.h>
#include <cctype.h> // include la libreria per manipolare i caratteri
#include <string.h> // include la libreria per manipolare le stringhe
#include <time.h> // libreria per randomizzare il seed

#define LUNG_NOME 20 // lunghezza massima del nome
#define LUNG_COGNOME 20 // lunghezza massima del cognome
#define LUNG_SIMBOLI 5 // vocabolario dei simboli (estendibile)

#define LUNG_PASSWORD 9 // lunghezza della password (aggiungere il terminatore!)
#define NUM_CAR_NOME_PASSWORD 3 // numero di caratteri estratti dal nome da inserire nella password
#define NUM_CAR_COGNOME_PASSWORD 3 // numero di caratteri del cognome da inserire nella password
#define NUM_SYMBOLI_PASSWORD 2 // numero di simboli nella password

// PROTOTIPO DI FUNZIONE: input -> nome e cognome, output -> password
char* generaPassword(char* nome, char* cognome);
```

29/04/19

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

83

## Esercizio 9.2 - Soluzione

```
int main(void) {
    char nome[LUNG_NOME]; // variabili locali per input e output
    char cognome[LUNG_COGNOME];
    char* password;

    printf("Inserisci nome: "); // inserimento input da tastiera
    scanf("%20s", nome);
    printf("\nInserisci cognome: ");
    scanf("%20s", cognome);

    password = generaPassword(nome, cognome);
    // genero la password e la mostro in output

    printf("\n\nCiao, %s. La password generata per te è: %s",
           nome, cognome, password);
}
```

29/04/19

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

84

## Esercizio 9.2 - Soluzione

```
char* generaPassword(char* nome, char* cognome) {

    char simboli[LUNG_SIMBOLI] = {'$', '#', '*', '^', '%'}; // estendibile
    int seed = time(NULL);
    srand(seed); // randomizzo la generazione

    // alloco un numero sufficiente di caratteri per generare la nuova password
    char* p = (char*) calloc(LUNG_PASSWORD, sizeof(char));

    // i primi caratteri della password sono estratti dal cognome. Utilizzo l'aritmetica dei puntatori per scorrere
    // tra i caratteri, i = contatore che scorre tra gli elementi del cognome, j = contatore che scorre tra gli
    // elementi della password (viene inizializzata in base al numero di caratteri estratti dal nome contenuti
    // nella password)
    int j = NUM_CAR_NOME_PASSWORD;
    for(int i=0; i < NUM_CAR_COGNOME_PASSWORD; i++) {
        *(p+j) = tolower( *(cognome+i) );
        j++; // incremento il contatore per puntare all'elemento successivo
    }

    int k = NUM_CAR_NOME_PASSWORD + NUM_CAR_COGNOME_PASSWORD;
    for(int i=0; i < NUM_SYMBOLI_PASSWORD; i++) {
        int idSimbolo = rand() % LUNG_SIMBOLI;
        *(p+k) = simboli[idSimbolo];
        k++;
    }

    return p; // restituisco la password generata
}
```

29/04/19

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

85

## Esercizio 9.2 - Soluzione

```
// i primi caratteri della password sono estratti dal cognome. Utilizzo l'aritmetica dei puntatori per scorrere
// tra i caratteri, i = contatore che scorre tra gli elementi del cognome, j = contatore che scorre tra gli
// elementi della password (viene inizializzata in base al numero di caratteri estratti dal nome contenuti
// nella password)

int j = NUM_CAR_NOME_PASSWORD;
for(int i=0; i < NUM_CAR_COGNOME_PASSWORD; i++) {
    *(p+j) = tolower( *(cognome+i) );
    j++; // incremento il contatore per puntare all'elemento successivo
}

int k = NUM_CAR_NOME_PASSWORD + NUM_CAR_COGNOME_PASSWORD;
for(int i=0; i < NUM_SYMBOLI_PASSWORD; i++) {
    int idSimbolo = rand() % LUNG_SIMBOLI;
    *(p+k) = simboli[idSimbolo];
    k++;
}

return p; // restituisco la password generata
}
```

29/04/19

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

86

## Esercizio 9.2 - Soluzione

```
// i primi caratteri della password sono estratti dal cognome. Utilizzo l'aritmetica dei puntatori per scorrere
// tra i caratteri, i = contatore che scorre tra gli elementi del cognome, j = contatore che scorre tra gli
// elementi della password (viene inizializzata in base al numero di caratteri estratti dal nome contenuti
// nella password)

int j = NUM_CAR_NOME_PASSWORD;
for(int i=0; i < NUM_CAR_COGNOME_PASSWORD; i++) {
    *(p+j) = tolower( *(cognome+i) );
    j++; // Incremento il contatore per puntare all'elemento successivo
}

int k = NUM_CAR_NOME_PASSWORD + NUM_CAR_COGNOME_PASSWORD;

for(int i=0; i < NUM_SIMBOLI_PASSWORD; i++) {
    int idSimbolo = rand() % LUNG_SIMBOLI;

    *(pk) = simboli[idSimbolo];
    k++;
}

return p; // restituisco la password generata
}
```

Che bug può avere  
questo codice?

29/04/19

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

87

## Esercizio 9.2 - Soluzione

```
// i primi caratteri della password sono estratti dal cognome. Utilizzo l'aritmetica dei puntatori per scorrere
// tra i caratteri, i = contatore che scorre tra gli elementi del cognome, j = contatore che scorre tra gli
// elementi della password (viene inizializzata in base al numero di caratteri estratti dal nome contenuti
// nella password)

int j = NUM_CAR_NOME_PASSWORD;
for(int i=0; i < NUM_CAR_COGNOME_PASSWORD; i++) {
    *(p+j) = tolower( *(cognome+i) );
    j++; // Incremento il contatore per puntare all'elemento successivo
}

int k = NUM_CAR_NOME_PASSWORD + NUM_CAR_COGNOME_PASSWORD;

for(int i=0; i < NUM_SIMBOLI_PASSWORD; i++) {
    int idSimbolo = rand() % LUNG_SIMBOLI;

    *(pk) = simboli[idSimbolo];
    k++;
}

return p; // restituisco la password generata
}
```

Se la lunghezza del nome o del  
cognome è inferiore del numero  
di caratteri da estrarre, possiamo  
avere comportamenti inattesi!  
**Miglioriamo la programmazione  
difensiva**

29/04/19

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

88

## Esercizio 9.2 – Soluzione Estesa (Parziale)

```
int car_nome = 0; // variabile di controllo per verificare la lunghezza del nome
int car_cognome = 0;

// verifico che il nome sia lungo a sufficienza, altrimenti lo inserisco tutto
if( strlen(nome) < NUM_CAR_NOME_PASSWORD )
    car_nome = strlen(nome);
else
    car_nome = NUM_CAR_NOME_PASSWORD;

// i primi caratteri della password sono estratti dal nome.
// Utilizzo l'aritmetica dei puntatori per scorrere tra i caratteri
// utilizza la funzione tolower() per convertire tutto in minuscolo
for(int i=0; i < car_nome; i++) {
    *(p+i) = tolower( *(nome+i) );
}
```

29/04/19

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

89

## Esercizio 9.3

- Estendere l'Esercizio 9.2 scrivendo una funzione che offuschi la password generata dall'utente, visualizzando soltanto i primi 2 e gli ultimi 2 caratteri.
  - Input: catmus\$#
  - Output: ca\*\*\*\*\$#
- Riflettere sul prototipo della funzione
  - Che opzioni abbiamo?
    - char\* offuscaPassword(char\* password)

29/04/19

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

90

## Esercizio 9.3

- Estendere l'Esercizio 9.2 scrivendo una funzione che offuschi la password generata dall'utente, visualizzando soltanto i primi 2 e gli ultimi 2 caratteri.

- Input: catmus\$#
- Output: ca\*\*\*\*\*\$#

### Riflettere sul prototipo della funzione

- Che opzioni abbiamo?
  - char\* offuscaPassword(char\* password)
- Oppure
  - void offuscaPassword(char\* password)

29/04/19

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

91

## Esercizio 9.3

- Estendere l'Esercizio 9.2 scrivendo una funzione che offuschi la password generata dall'utente, visualizzando soltanto i primi 2 e gli ultimi 2 caratteri.

- Input: catmus\$#
- Output: ca\*\*\*\*\*\$#

### Riflettere sul prototipo della funzione

- Che opzioni abbiamo?
  - char\* offuscaPassword(char\* password)
- Oppure
  - void offuscaPassword(char\* password)

Sono entrambe corrette! La prima è più comprensibile e dà maggiormente l'idea di «restituire» un valore differente (offuscato). La seconda è comunque corretta, ma MODIFICA il valore originale della password! Bisogna valutare gli effetti collaterali

29/04/19

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

92

## Esercizio 9.3 – Soluzione (Parziale)

```
/* La funzione prende in input una password e la offusca, mostrando solo i primi due e gli ultimi due caratteri
char* offuscaPassword(char* password) {
    // alloco un numero sufficiente di caratteri per generare la nuova password
    char* p = (char*) calloc(LUNG_PASSWORD, sizeof(char));
    int lunghezza = strlen(password); // memorizzo la lunghezza della password

    for(int i=0; i < CAR_INIZIALI_OFFUSCATI; i++) {
        *(p+i) = *(password+i); // copio i primi caratteri
    }

    for(int i=CAR_INIZIALI_OFFUSCATI; i < lunghezza - CAR_FINALI_OFFUSCATI; i++) {
        *(p+i) = '*'; // offuso i caratteri centrali
    }

    for(int i=lunghezza - CAR_FINALI_OFFUSCATI; i < lunghezza; i++) {
        *(p+i) = *(password+i); // copio gli ultimi caratteri
    }

    return p; // restituisco la password offuscata
}
```

29/04/19

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

93

## Puntatori Costanti

- Come visto nell'esercizio 9.3, quando passiamo un puntatore possiamo **modificare il contenuto puntato**

- Questo potrebbe essere un comportamento non desiderato

29/04/19

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

94

## Puntatori Costanti

- Come visto nell'esercizio 9.3, quando passiamo un puntatore possiamo **modificare il contenuto puntato**
  - Questo potrebbe essere un comportamento non desiderato
- **Preferiamo sempre il passaggio per valore**
  - In alcuni casi questo non è possibile
    - Array
  - In altri casi, questo non è conveniente
    - Strutture di dimensioni molto elevate

29/04/19

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

95

## (torniamo a...) Puntatori Costanti

- Come visto nell'esercizio 9.3, quando passiamo un puntatore possiamo **modificare il contenuto puntato**
  - Questo potrebbe essere un comportamento non desiderato
- **Preferiamo sempre il passaggio per valore**
  - In alcuni casi questo non è possibile
    - Array
  - In altri casi, questo non è conveniente
    - Strutture di dimensioni molto elevate
- Il modificatore **const** indica al compilatore di **controllare possibili accessi in scrittura al contenuto puntato**

29/04/19

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

96

## Puntatori Costanti

```
char* offuscaPassword(char* password)
{
    // implementazione
}

char* offuscaPassword(const char* password)
{
    // implementazione
}
```

Se siamo sicuri di non dover operare modifiche a un vettore (o più in generale, **a una qualsiasi variabile presente nel codice**), utilizziamo il quantificatore **const** per indicare al compilatore che il valore di quella variabile non deve essere modificato durante l'esecuzione.  
Utile a prevenire comportamenti imprevedibili nel codice e a ridurre i bug.

29/04/19

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

97

## Puntatori Costanti

- Quando passiamo un puntatore, possiamo **modificare il contenuto puntato**
  - Questo potrebbe essere un comportamento non desiderato
- **Preferiamo sempre il passaggio per valore**
  - In alcuni casi questo non è possibile
    - Array
  - **In altri casi, questo non è conveniente**
    - Strutture di dimensioni molto elevate

**Approfondiamo.**



29/04/19

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

98

## Passaggio di Parametri con le **struct**

- Le variabili tradizionali vengono normalmente passate per valore.
- **Gli array vengono passati per riferimento**
- **...e le struct?**

29/04/19

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

99

## Passaggio di Parametri con le **struct**

- Le variabili tradizionali vengono normalmente passate per valore.
- **Gli array vengono passati per riferimento**
- **...e le struct?**
  - Di default le **struct** vengono passate per valore.
  - Nel passaggio per valore viene effettuata **una copia dell'intera struttura**.

29/04/19

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

100

## Passaggio di Parametri con le **struct**

- Le variabili tradizionali vengono normalmente passate per valore.
- **Gli array vengono passati per riferimento**
- **...e le struct?**
  - Di default le **struct** vengono passate per valore.
  - Nel passaggio per valore viene effettuata **una copia dell'intera struttura**.
- **Problema**
  - Per strutture particolarmente grandi, **l'operazione di copia può essere dispendiosa**
  - Il passaggio per riferimento può essere un'alternativa più efficiente.
    - Si passa l'**indirizzo della struct** alla funzione
    - Bisogna stare attenti a non modificare accidentalmente i valori, perché eventuali modifiche sarebbero ereditate dalla funzione chiamante

29/04/19

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

101

## Passaggio di Parametri con le **struct** - Esempio

```

1 // Laboratorio di Informatica
2 // Informatica - TPS - 2016/2017
3 // docente: Cataldo Musto
4
5 #include <stdio.h>
6
7 int main() {
8
9 // Dichiaro una struct di tipo "data"
10 typedef struct {
11     int giorno;
12     char mese[15];
13     int anno;
14 } data ;
15
16 data d1;
17 data d2;
18

```

29/04/19

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

102

## Passaggio di Parametri con le **struct** - Esempio

```

1 // Laboratorio di Informatica
2 // Informatica - TPS - 2016/2017
3 // docente: Cataldo Musto
4
5 #include <stdio.h>
6
7~ int main() {
8
9 // Dichiaro una struct di tipo "data"
10 typedef struct {
11     int giorno;
12     char mese[15];
13     int anno;
14 } data ;
15
16 data d1;
17 data d2;
18

```

Qual è la dimensione di questa **struct**?  
24 byte (4 + 15 + 4 + 1 padding)

29/04/19

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

103

## Passaggio di Parametri con le **struct** - Esempio

```

1 // Laboratorio di Informatica
2 // Informatica - TPS - 2016/2017
3 // docente: Cataldo Musto
4
5 #include <stdio.h>
6
7~ int main() {
8
9 // Dichiaro una struct di tipo "data"
10 typedef struct {
11     int giorno;
12     char mese[15];
13     int anno;
14 } data ;
15
16 data d1;
17 data d2;
18

```

// Prototipi di funzione  
void function\_data (data d);  
void function\_data\_ptr (data\* dPtr);

void function\_data (data d) {  
 ...  
}  
void function\_data\_ptr (data\* dPtr) {  
 ...  
}

// main  
int main() {  
 function\_data(d1);  
 function\_data\_ptr(&d2);  
}

29/04/19

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

104

## Passaggio di Parametri con le **struct** - Esempio

```

1 // Laboratorio di Informatica
2
3 Le funzioni sono identiche. La seconda però
4 effettua il passaggio per riferimento, quindi è più
5 efficiente. Bisogna stare attenti agli 'effetti
6 collaterali'
7~ int main() {
8
9 // Dichiaro una struct di tipo "data"
10 typedef struct {
11     int giorno;
12     char mese[15];
13     int anno;
14 } data ;
15
16 data d1;
17 data d2;
18

```

// Prototipi di funzione  
void function\_data (data d);  
void function\_data\_ptr (data\* dPtr);

void function\_data (data d) {  
 ...  
}  
void function\_data\_ptr (data\* dPtr) {  
 ...  
}

// main  
int main() {  
 function\_data(d1);  
 function\_data\_ptr(&d2);  
}

29/04/19

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

105

## Note Conclusive

### Puntatori

- Particolare tipologia di variabili. Possono memorizzare solo **indirizzi di memoria**. Si inizializzano a **NULL**.
- Per risalire all'indirizzo di una variabile si utilizza l'**operatore indirizzo (&)**
- Per risalire al valore puntato da un indirizzo si usa l'**operatore di indirezione (\*)**

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

106

## Note Conclusive

- **Puntatori**

- Particolare tipologia di variabili. Possono memorizzare solo **indirizzi di memoria**. Si inizializzano a **NULL**.
- Per risalire all'indirizzo di una variabile si utilizza l'**operatore indirizzo (&)**
- Per risalire al valore puntato da un indirizzo si usa l'**operatore di indirezione (\*)**

- **Applicazioni**

- **Passaggio dei parametri per riferimento.** Si passa a una funzione l'indirizzo di una variabile, invece del suo valore.
- **Obbligatorio per gli array** (un array è già un puntatore al primo elemento)
- **Spesso utile per le struct** (motivi di efficienza)
- **Valutare l'uso per le altre tipologie di variabili**
  - Se la funzione ha bisogno di modificare la variabile (es. scambio valori), è necessario passare il parametro per indirizzo. Altrimenti è sufficiente il passaggio per valore (più sicuro.)

29/04/19

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

107

## Note Conclusive (cont.)

- **Aritmetica dei Puntatori**

- I puntatori possono anche essere usati dentro espressioni matematiche
- Ci forniscono una modalità alternativa per accedere e modificare i valori negli array
- *Imparare l'aritmetica dei puntatori: possibili esercizi.*

29/04/19

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

108

## Note Conclusive (cont.)

- **Aritmetica dei Puntatori**

- I puntatori possono anche essere usati dentro espressioni matematiche
- Ci forniscono una modalità alternativa per accedere e modificare i valori negli array
- *Imparare l'aritmetica dei puntatori: possibili esercizi.*

- **Utilizzi Avanzati**

- I puntatori possono essere inseriti a loro volta dentro **array** e dentro **struct**
- **Applicazione:** **array di stringhe == array di puntatori a char**
- **Se una funzione deve restituire una stringa, è necessario utilizzare malloc() e calloc()**
- Il quantificatore **const** serve ad aiutarci ad evitare modifiche non necessarie alle variabili. Utile quando preferiamo utilizzare i puntatori per motivi di efficienza, ma vogliamo evitare modifiche accidentali. **Utile adottarlo per identificare le variabili che non saranno modificate!**

29/04/19

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

109

**DOMANDE?**



29/04/19

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

110