Esercitazione 07

2025/10/22 – Puntatori

L. Alessandrini

Credits: A. Montenegro per gli esercizi

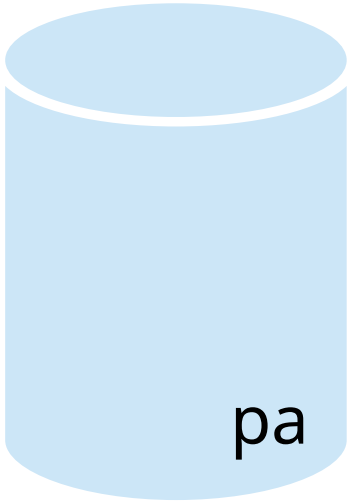
Puntatori

Funzionamento Base

Puntatori

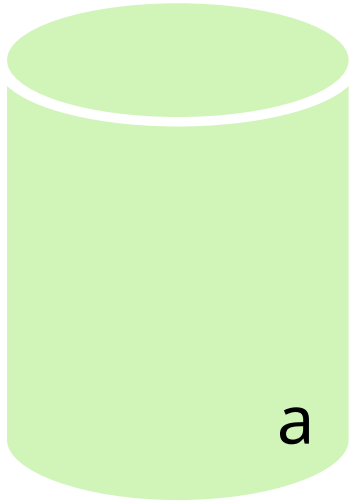
Funzionamento Base

```
int *pa;
```



0x7ff7bfeff310

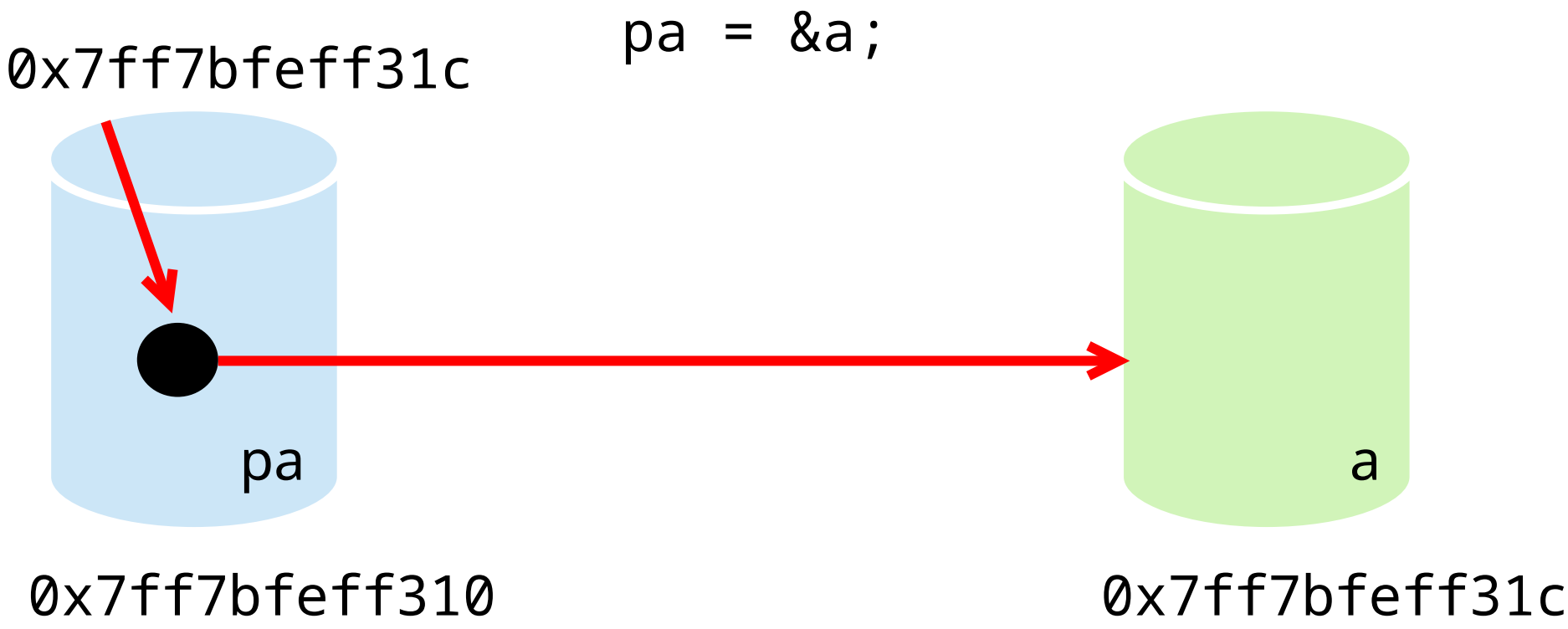
```
int a;
```



0x7ff7bfeff31c

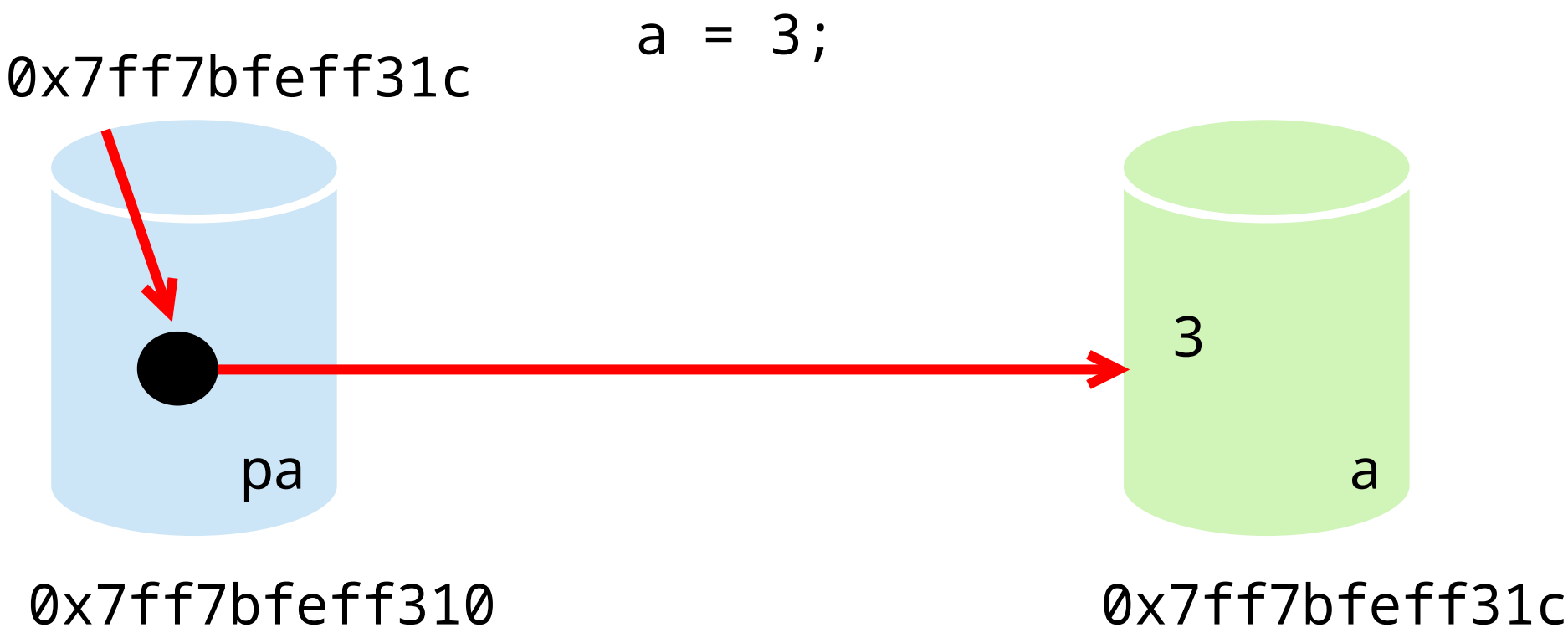
Puntatori

Funzionamento Base



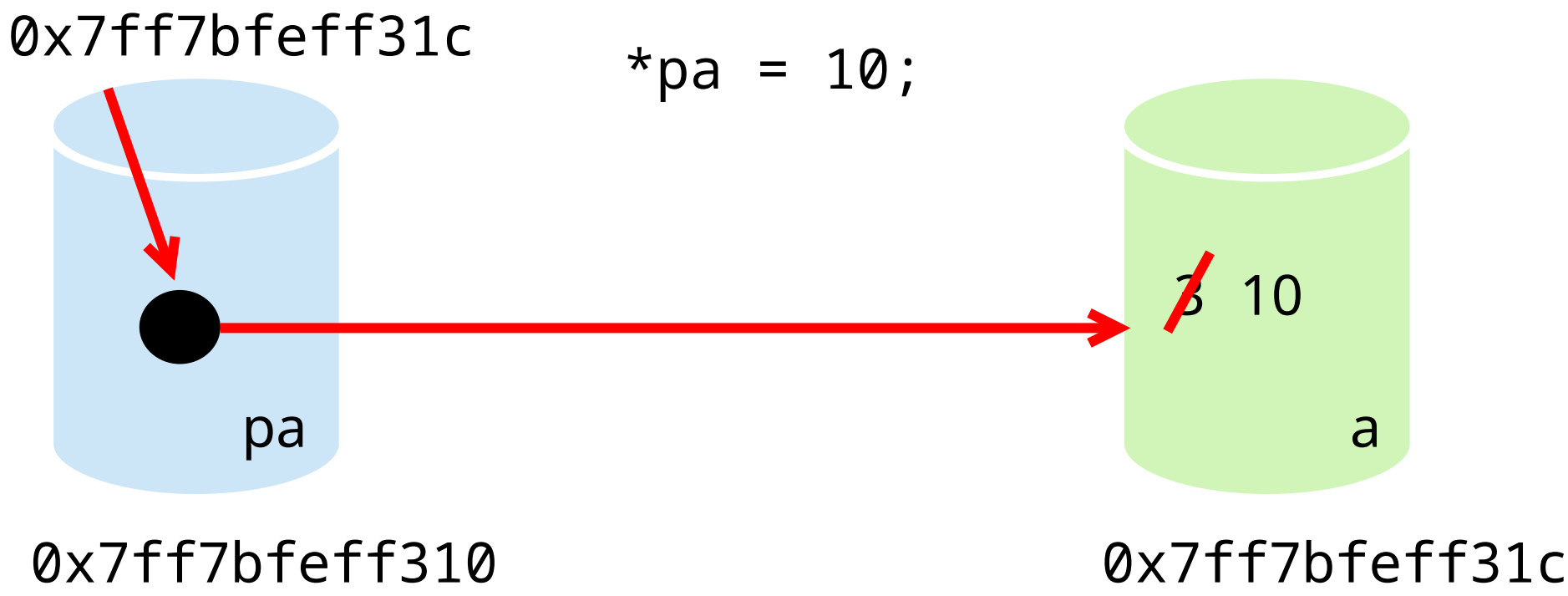
Puntatori

Funzionamento Base



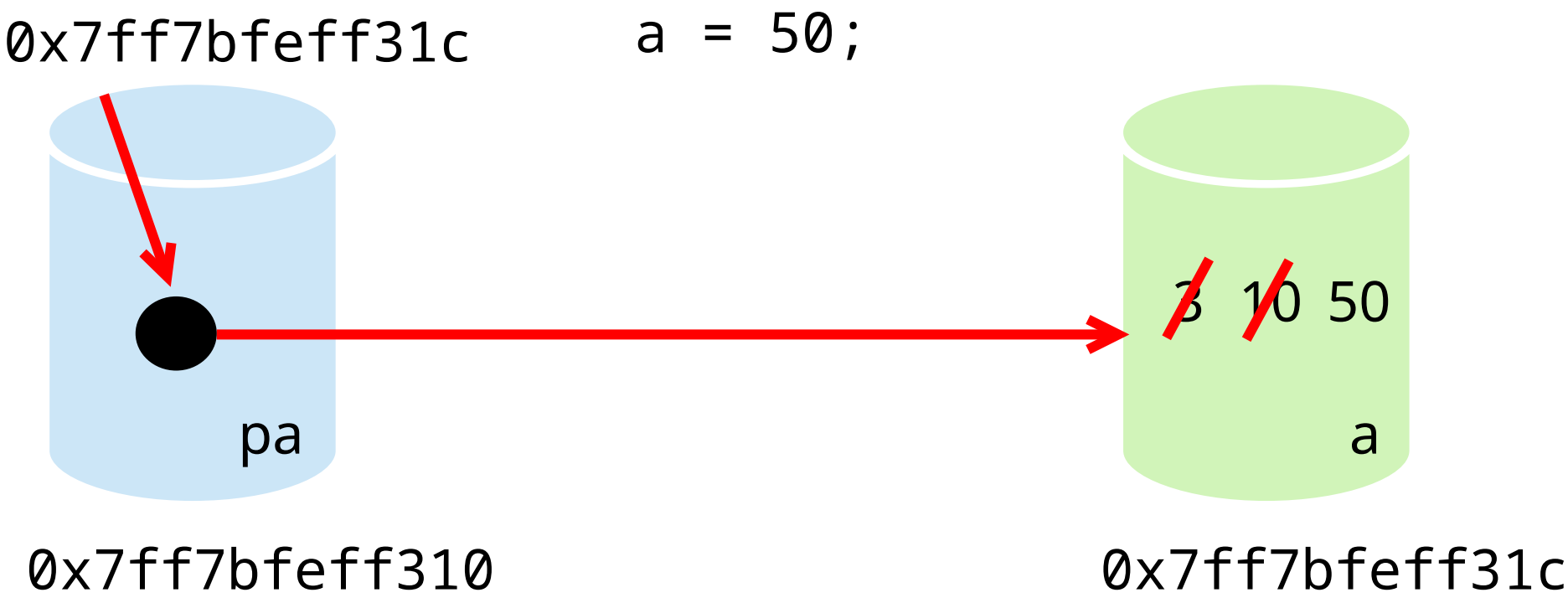
Puntatori

Funzionamento Base



Puntatori

Funzionamento Base



Puntatori

Esercizi – Parte 1

Esercizio 01

Utilizzo Base dei puntatori

Testo

Scrivi un programma che utilizza un puntatore per leggere dallo standard input un numero intero inserito dall’utente. Il programma raddoppia il valore inserito dall’utente utilizzando il puntatore. Stampare, prima e dopo il raddoppio del valore, le seguenti informazioni:

1. Indirizzo della variabile puntata dal puntatore
2. Indirizzo della variabile a cui punta il puntatore
3. Valore della variabile
4. Valore della variabile a cui punta il puntatore

Stampare indirizzo di memoria

%p

Per stampare l’indirizzo di memoria è sufficiente usare il modificatore %p, come nella prossima slide[1]

1. Credits, anche per il codice della prossima slide: https://www.w3schools.com/c/c_pointers.php

Esercizio 02

Swap

Testo

Scrivi un programma che implementa il meccanismo di **swap** tra due puntatori, che scambia i valori di due variabili intere usando i puntatori. Il programma dovrà:

1. Leggere due interi da input.
2. Usare il meccanismo di **swap** per scambiare i valori.
3. Stampare i valori prima e dopo lo scambio.

Quanto occupa un puntatore?

Dimensioni

Considerate il seguente codice

puntatore?

```c
#include <stdio.h>

// STAMPO DIMENSIONI DI
DIVERSI PUNTATORI
int main() {
    int a, *pInt;
    char b, *pChar;
    double c, *pDouble;
    float d, *pFloat;

    typedef struct {
        int a;
        char b;
        double c;
    } MyStruct;

    MyStruct s, *pStruct;
```

```c
    s.a=10;
    s.b='z';
    s.c=3.15;

    // inizializzazione
    // puntatori
    pInt = &a;
    pChar = &b;
    pDouble = &c;
    pFloat = &d;
    pStruct = &s;
```

```c
    // sizeof restituisce la dimensione in byte del tipo di dato
    // source: https://learn.microsoft.com/it-it/cpp/c-language/sizeof-operator-c?view=msvc-170

    printf("Size of int pointer: %d bytes\n", sizeof(pInt));
    printf("Size of char pointer: %d bytes\n", sizeof(pChar));
    printf("Size of double pointer: %d bytes\n", sizeof(pDouble));
    printf("Size of float pointer: %d bytes\n", sizeof(pFloat));
    printf("Size of MyStruct pointer: %d bytes\n", sizeof(pStruct));
        int a;
        char b;
        double c;
    } MyStruct;

    MyStruct s, *pStruct;
```

```c
    pChar = &b
    pDouble = &
    pFloat = &c;
    pStruct = &s;
```

Cosa ci aspettiamo?

```c
    // sizeof restituisce la dimensione in byte del tipo di dato
    // source: https://learn.microsoft.com/it-it/cpp/c-language/sizeof-operator-c?view=msvc-170

    printf("Size of int pointer: %d bytes\n", sizeof(pInt));
    printf("Size of char pointer: %d bytes\n", sizeof(pChar));
    printf("Size of double pointer: %d bytes\n", sizeof(pDouble));
    printf("Size of float pointer: %d bytes\n", sizeof(pFloat));
    printf("Size of MyStruct pointer: %d bytes\n", sizeof(pStruct));
```

```c
        int a;
        char b;
        doub
    } MyStru

    MyStruct
```

```c
pChar = &b
&s;
```

```
Size of int pointer: 8 bytes
Size of char pointer: 8 bytes
Size of double pointer: 8 bytes
Size of float pointer: 8 bytes
Size of MyStruct pointer: 8 bytes
```

Cosa ci aspettiamo?

```
    // sizeof restituisce la dimensione in byte del tipo di dato
    // source: https://learn.microsoft.com/it-it/cpp/c-
language/sizeof-operator-c?view=msvc-170

    printf("Size of int pointer: %d bytes\n", sizeof(pInt));
    printf("Size of char pointer: %d bytes\n", sizeof(pChar));
    printf("Size of double pointer: %d bytes\n",
sizeof(pDouble));
    printf("Size of float pointer: %d bytes\n", sizeof(pFloat));
    printf("Size of MyStruct pointer: %d bytes\n",
sizeof(pStruct));
        int a;
        char b;
        doub
    } MyStru

    MyStruct
```

```
Size of int pointer: 8 bytes
Size of char pointer: 8 bytes
Size of double pointer: 8 bytes
Size of float pointer: 8 bytes
Size of MyStruct pointer: 8 bytes
```

Cosa ci aspettiamo?

N.B.: architettura a 64 bit

puntatore?

Cosa ci aspettiamo?

```c
#include <stdio.h>

// STAMPO DIMENSIONI DI
DIVERSI PUNTATORI
i
    // Dimensione dei puntati
    printf("Size of int: %d bytes\n", sizeof(*pInt));
    printf("Size of char: %d bytes\n", sizeof(*pChar));
    printf("Size of double: %d bytes\n", sizeof(*pDouble));
    printf("Size of float: %d bytes\n", sizeof(*pFloat));
    printf("Size of MyStruct: %d bytes\n", sizeof(*pStruct));

    return 0;
}
        double c;
    } MyStruct;

    MyStruct s, *pStruct;
```

```c
    pFloat = &d;
    pStruct = &s;
```

puntatore?

Cosa ci aspettiamo?

```c
#include <stdio.h>

// STAMPO DIMENSIONI DI DIVERSI PUNTATORI
i
    // Dimensione dei puntati
    printf("Size of int: %d bytes\n", sizeof(*pInt));
    printf("Size of char: %d bytes\n", sizeof(*pChar));
    printf("Size of double: %d bytes\n", sizeof(*pDouble));
    printf("Size of float: %d bytes\n", sizeof(*pFloat));
    printf("Size of MyStruct: %d bytes\n", sizeof(*pStruct));

    return 0;
}
        double c
    } MyStruct;

    MyStruct s,
```

```
Size of int: 4 bytes
Size of char: 1 bytes
Size of double: 8 bytes
Size of float: 4 bytes
Size of MyStruct: 16 bytes
```

```c
&d;
 &s;
```

Se incrementassimo i puntatori?

Considerate il seguente codice

```c
#include <stdio.h>
int main() {
    int var = 20;   // variabile intera
    int *p;         // puntatore a intero che verrà usato per l'incremento
    int *p_start;   // puntatore a intero per memorizzare l'indirizzo iniziale
    char *pc;       // puntatore a char che verrà usato per l'incremento
    char *pc_start;   // puntatore a char per memorizzare l'indirizzo iniziale
    char cvar = 'A';   // variabile char

    printf("Dimensioni: sizeof(int)=%d bytes, sizeof(char)=%d bytes\n",
sizeof(int), sizeof(char));

    p = &var;      // assegnazione indirizzo di var al puntatore p
    p_start = &var;   // memorizzazione dell'indirizzo iniziale
    pc = &cvar;    // assegnazione indirizzo di cvar al puntatore pc
    pc_start = &cvar; // memorizzazione dell'indirizzo iniziale
```

```
    printf("##### CASO INTERO #####\n");
    printf("Indirizzo iniziale del puntatore a intero: %p\n", p); //
0x7fffffffe1b8, i primi sono a zero
    p++;  // incremento del puntatore a intero
    printf("Indirizzo del puntatore a intero dopo incremento: %p\n", p);
    // devo moltiplicare per sizeof int, altrimenti ritorna il numero di
posizioni incrementate
    // La dimostrazione ne è la stampa delle posizioni
    printf("Stampo la differenza come un indirizzo: %p \n", (p -
p_start)*sizeof(int));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n", p -
p_start);
```

Cosa ci aspettiamo?

```
    printf("##### CASO INTERO #####\n");
    printf("Indirizzo iniziale del puntatore a intero: %p\n", p); //
0x7ffffffe1b8, i primi sono a zero
    p++;  // incremento del puntatore a intero
    printf("Indirizzo del puntatore a intero dopo incremento: %p\n", p);
    // devo moltiplicare per sizeof int, altrimenti ritorna il numero di
posizioni incrementate
    // La dimostrazione ne è la stampa delle posizioni
    printf("Stampo la differenza come un indirizzo: %p \n", (p -
p_start)*sizeof(int));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n", p -
p_start);
```

Cosa ci

```
Dimensioni: sizeof(int)=4 bytes, sizeof(char)=1 bytes
##### CASO INTERO #####
Indirizzo iniziale del puntatore a intero: 0x7ffffffe1b4
Indirizzo del puntatore a intero dopo incremento: 0x7ffffffe1b8
Stampo la differenza come un indirizzo: 0x4
Stampo la differenza come un intero (num. posizioni): 1
```

```
    printf("##### CASO INTERO #####\n");
    printf("Indirizzo iniziale del puntatore a intero: %p\n", p); //
0x7fffffffe1b8, i primi sono a zero
    p++;  // incremento del puntatore a intero
    printf("I                ntatore a intero dopo incremento: %p\n", p);
    // devo m          sizeof int, altrimenti ritorna il numero di
posizioni inc
    // La dim                la stampa delle posizioni
    printf("Stampo la differenza come un indirizzo: %p \n", (p -
p_start)*sizeof(int));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n", p -
p_start);
```

Dimensione intero: 4 bytes

Cosa ci

```
Dimensioni: sizeof(int)=4 bytes, sizeof(char)=1 bytes
##### CASO INTERO #####
Indirizzo iniziale del puntatore a intero: 0x7fffffffe1b4
Indirizzo del puntatore a intero dopo incremento: 0x7fffffffe1b8
Stampo la differenza come un indirizzo: 0x4
Stampo la differenza come un intero (num. posizioni): 1
```

```
    printf("##### CASO INTERO #####\n");
    printf("Indirizzo iniziale del puntatore a intero: %p\n", p); //
0x7fffffffe1b8, i primi sono a zero
    p++;  // incremento del puntatore a
    printf("Indirizzo del puntatore a in                %p\n", p);
    // devo moltiplicare per sizeof int,              l numero di
posizioni incrementate
    // La dimostrazione ne è la stampa c
    printf("Stampo la differenza come un indirizzo: %p \n", (p -
p_start)*sizeof(int));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n", p -
p_start);
```

L'indirizzo, stampato con %p, è incrementato di 4 bytes!

Cosa ci

```
Dimensioni: sizeof(int)=4 bytes, sizeof(char)=1 bytes
##### CASO INTERO #####
Indirizzo iniziale del puntatore a intero: 0x7fffffffe1b4
Indirizzo del puntatore a intero dopo incremento: 0x7fffffffe1b8
Stampo la differenza come un indirizzo: 0x4
Stampo la differenza come un intero (num. posizioni): 1
```

```
    printf("##### CASO INTERO #####\n");
    printf("Indirizzo iniziale del punt
0x7fffffffe1b8, i primi sono a zero
    p++;  // incremento del puntatore a
    printf("Indirizzo del puntatore a i
    // devo moltiplicare per sizeof int
posizioni incrementate
    // La dimostrazione ne è la stampa delle posizioni
    printf("Stampo la differenza come un indirizzo: %p \n", (p -
p_start)*sizeof(int));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n", p -
p_start);
```

Questa ritornna il numero di posizioni incrementate!

Cosa ci

```
Dimensioni: sizeof(int)=4 bytes, sizeof(char)=1 bytes
##### CASO INTERO #####
Indirizzo iniziale del puntatore a intero: 0x7fffffffe1b4
Indirizzo del puntatore a intero dopo incremento: 0x7fffffffe1b8
Stampo la differenza come un indirizzo: 0x4
Stampo la differenza come un intero (num. posizioni): 1
```

```
    printf("##### CASO INTERO #####\n");
    printf("Indirizzo iniziale del punt
0x7fffffffe1b8, i primi sono a zero
    p++;  // incremento del puntatore a
    printf("Indirizzo del puntatore a i
    // devo moltiplicare per sizeof int
posizioni incrementate
    // la dimostrazione ne è la stampa delle posizioni
    printf("Stampo la differenza come un indirizzo: %p \n", (p -
p_start)*sizeof(int));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n", p -
p_start);
```

Questa ritornna il numero di posizioni incrementate!

Infatti, per ottenere i 4 bytes, devo moltiplicare per la dimensione dell'intero

Cosa ci

```
Dimensioni: sizeof(int)=4 bytes, sizeof(char)=1 bytes
##### CASO INTERO #####
Indirizzo iniziale del puntatore a intero: 0x7fffffffe1b4
Indirizzo del puntatore a intero dopo incremento: 0x7fffffffe1b8
Stampo la differenza come un indirizzo: 0x4
Stampo la differenza come un intero (num. posizioni): 1
```

```
    printf("\n##### CASO CHAR #####\n");
    printf("Indirizzo iniziale del puntatore a char: %p\n", pc);
    pc++;  // incremento del puntatore a char
    printf("Indirizzo del puntatore a char dopo incremento: %p\n", pc);
    printf("Stampo la differenza come un indirizzo: %p \n", (pc -
pc_start)*sizeof(char));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n",
pc - pc_start);
    return 0;
}
```

Cosa ci aspettiamo?

Ricorda: `sizeof(char)=1 bytes`

```
    printf("\n##### CASO CHAR #####\n");
    printf("Indirizzo iniziale del puntatore a char: %p\n", pc);
    pc++;  // incremento del puntatore a char
    printf("Indirizzo del puntatore a char dopo incremento: %p\n", pc);
    printf("Stampo la differenza come un indirizzo: %p \n", (pc -
pc_start)*sizeof(char));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n",
pc - pc_start);
    return 0;
}
```

Cosa ci aspettiamo?
Ricorda: `sizeof(char)=1 bytes`

```
##### CASO CHAR #####
Indirizzo iniziale del puntatore a char: 0x7fffffffe1b3
Indirizzo del puntatore a char dopo incremento: 0x7fffffffe1b4
Stampo la differenza come un indirizzo: 0x1
Stampo la differenza come un intero: 1
```

```
    printf("\n##### CASO CHAR #####\n");
    printf("Indirizzo iniziale del puntatore a char: %p\n", pc);
    pc++;  // incremento del puntatore a char
    printf("Indirizzo del puntatore a char dopo incremento: %p\n", pc);
    printf("Stampo la differenza come un indirizzo: %p \n", (pc -
pc_start)*sizeof(char));
    printf("Stampo la differenza come un intero (num. posizioni): %d \n",
pc - pc_start);
    return 0;
}
```

Nota l'incremento di un byte

```
##### CASO CHAR #####
Indirizzo iniziale del puntatore a char: 0x7fffffffe1b3
Indirizzo del puntatore a char dopo incremento: 0x7fffffffe1b4
Stampo la differenza come un indirizzo: 0x1
Stampo la differenza come un intero: 1
```

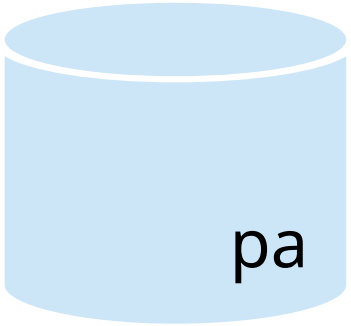
Puntatori e Array

Recap

Puntatori

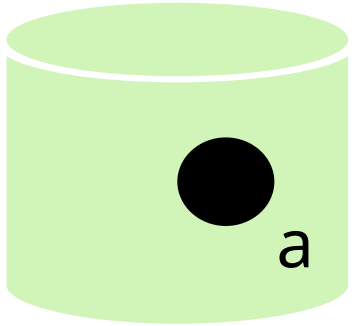
Puntatori e Array

int *pa;



0x7ff7bfeff310

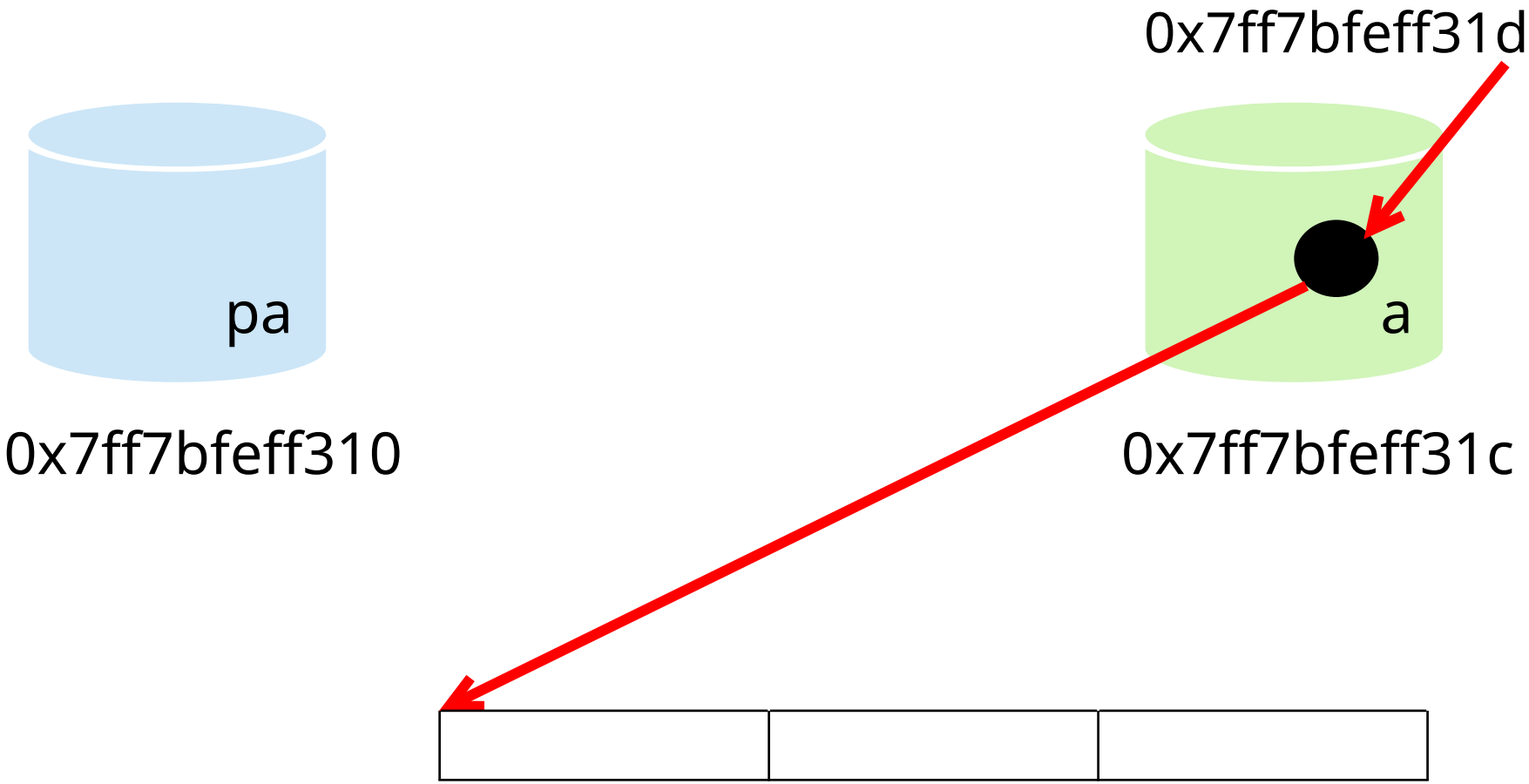
int a[3];



0x7ff7bfeff31c

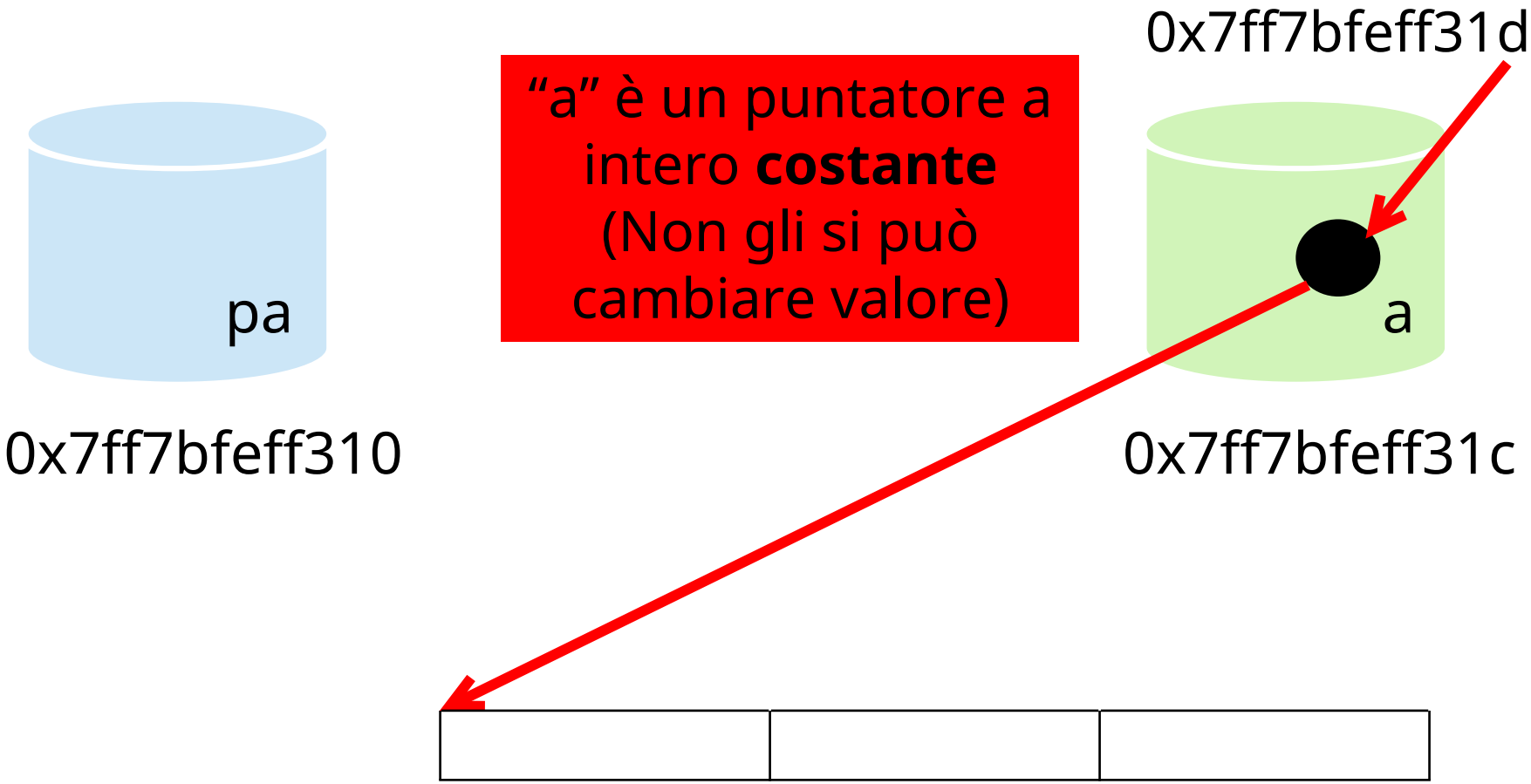
Puntatori

Puntatori e Array



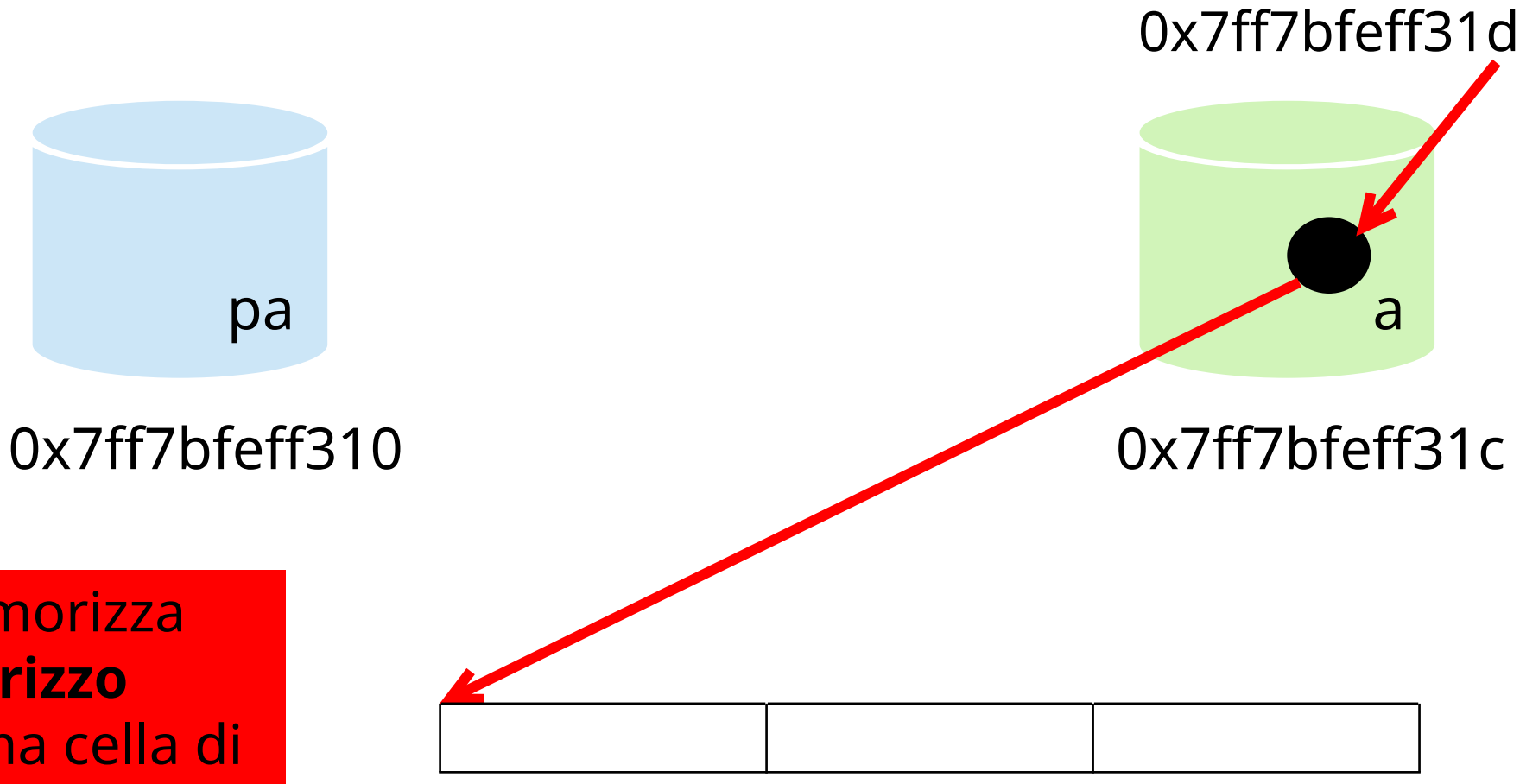
Puntatori

Puntatori e Array



Puntatori

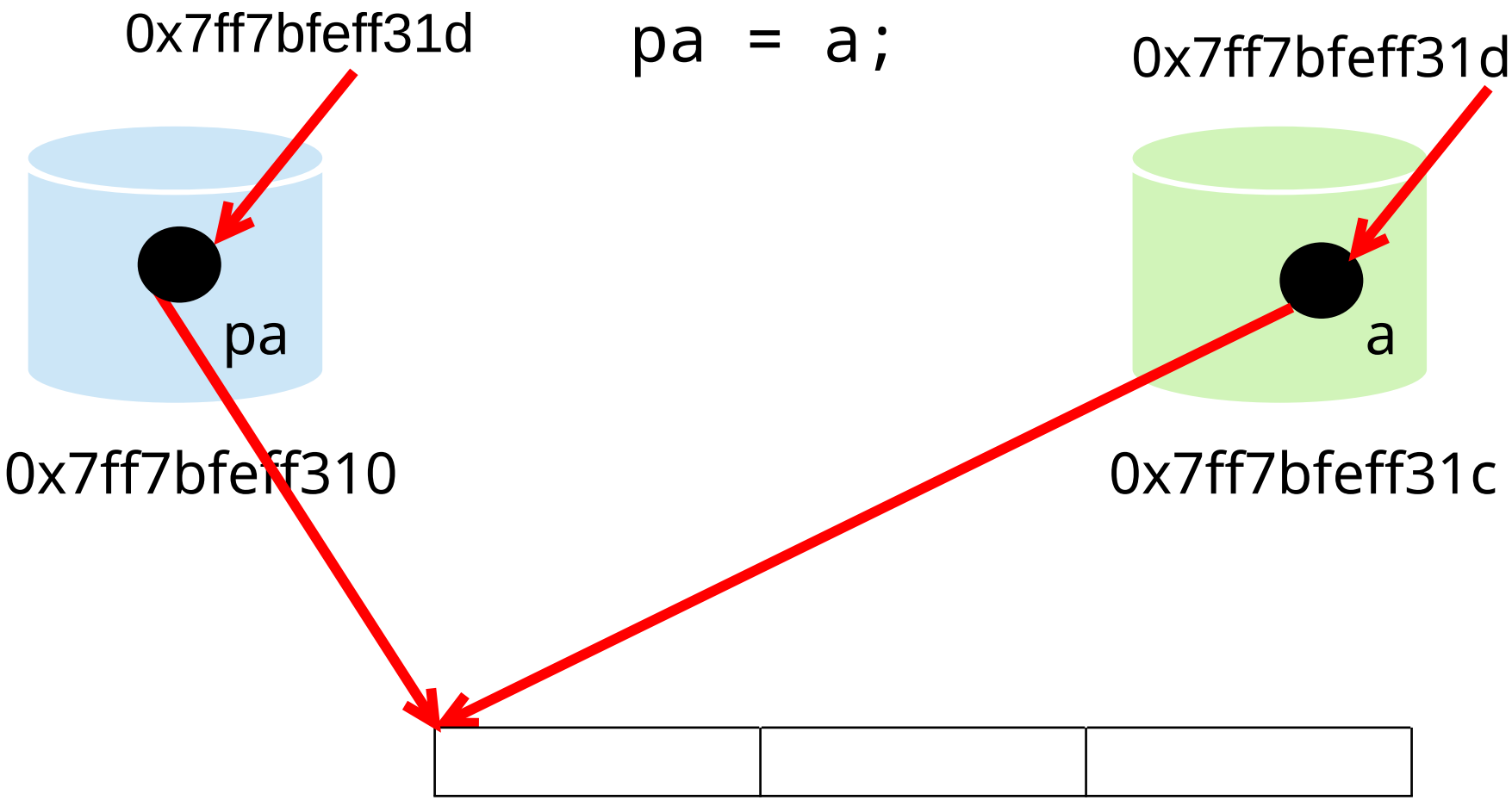
Puntatori e Array



"a" memorizza l'**indirizzo** della prima cella di memoria dell'array (`0x7ff7bfeff31d`)

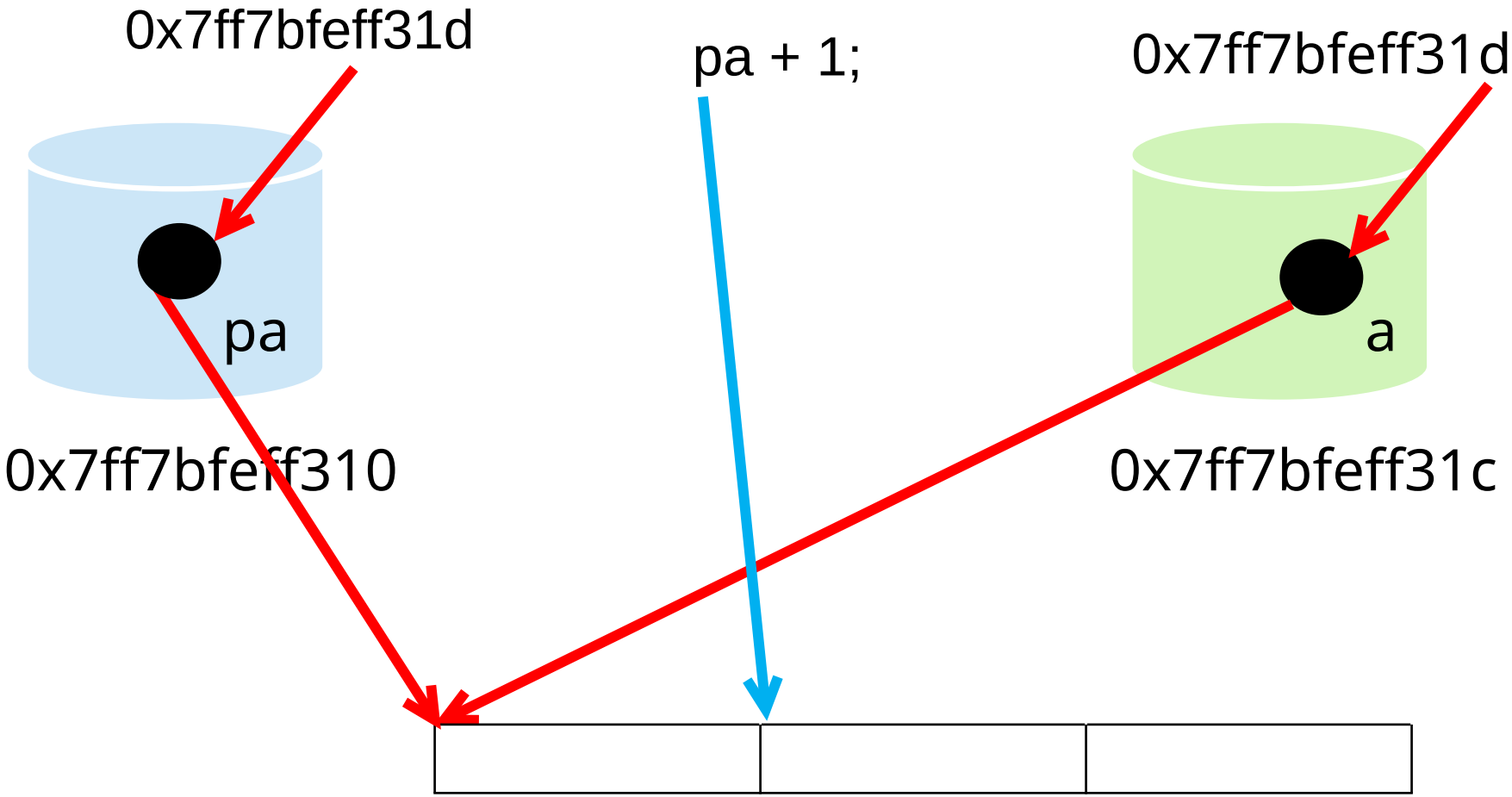
Puntatori

Puntatori e Array



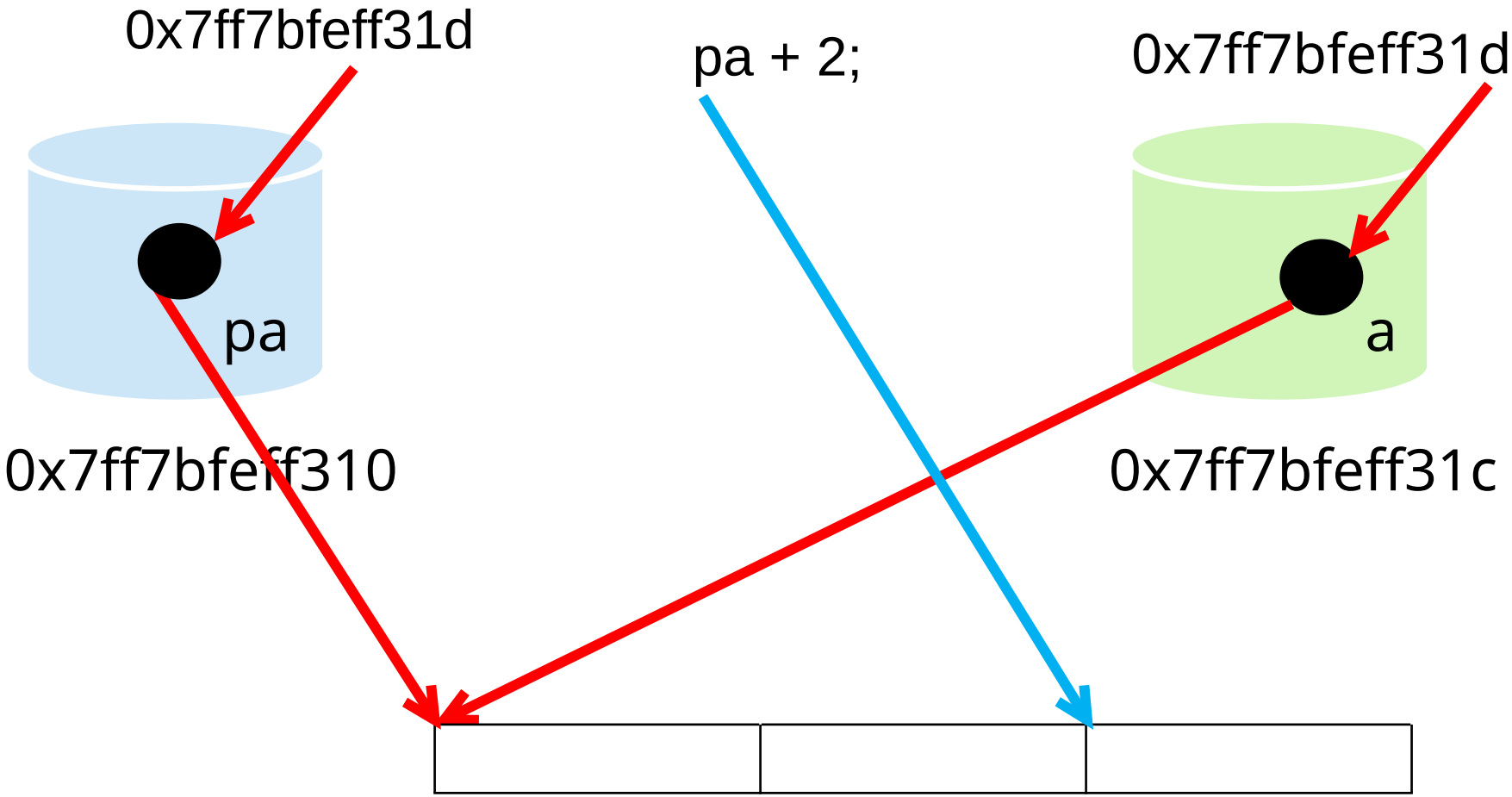
Puntatori

Puntatori e Array



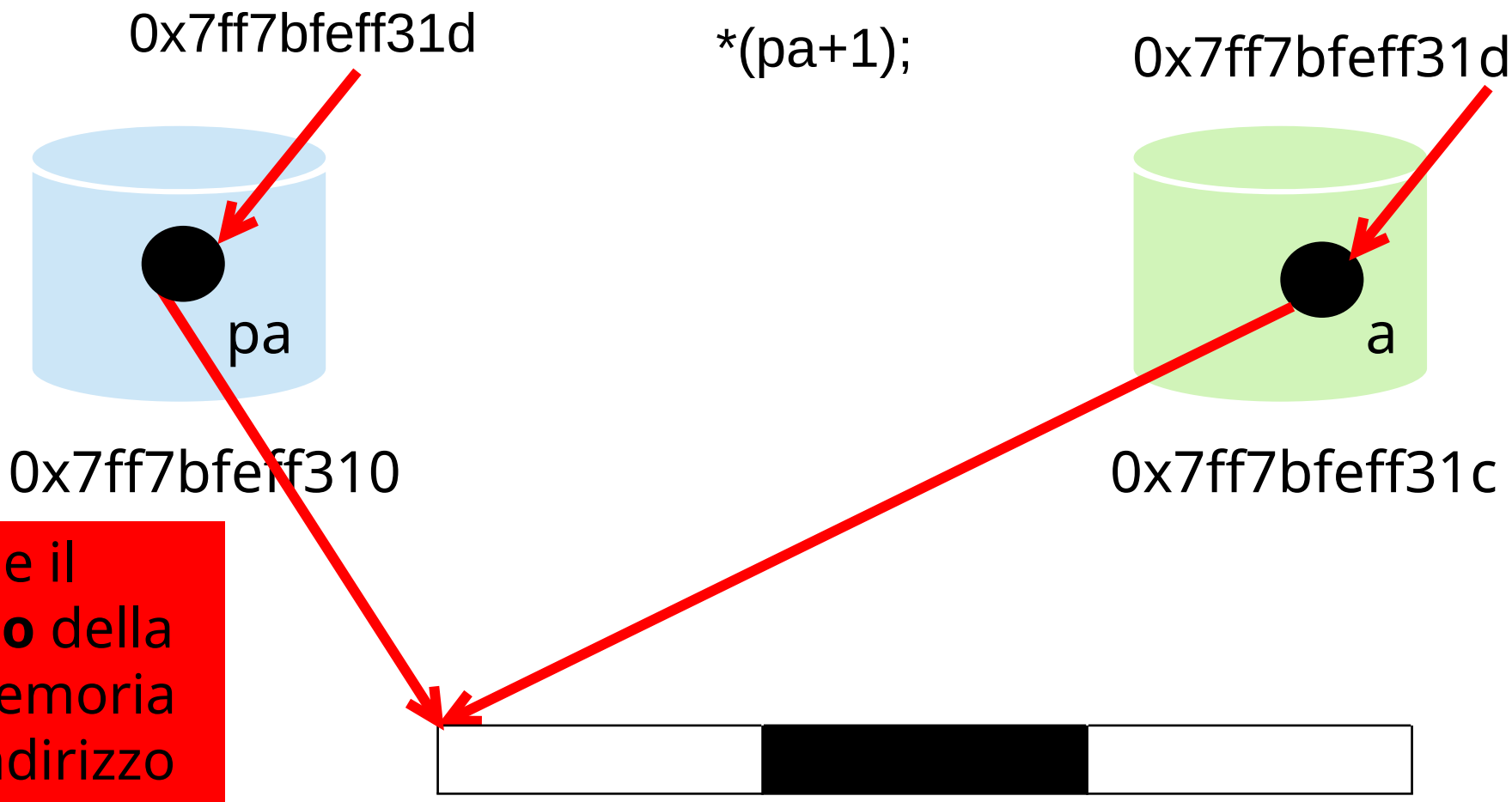
Puntatori

Puntatori e Array



Puntatori

Puntatori e Array



Prende il **contenuto** della cella di memoria che ha l'indirizzo pa+1
Come a[1]

Puntatori

Esercizi – Parte 2

Esercizio 03

Inserimento, Somma, Min e Max di Array

Testo

Scrivere un programma che dichiara un array di interi di 5 elementi e che usa un puntatore per:

1. Leggere i valori degli elementi dell’array dallo standard input
2. Sommare tutti gli elementi dell’array
3. Trovare il minimo valore dell’array
4. Trovare il massimo valore dell’array
5. Stampare la somma, il minimo e il massimo

Esercizio 04

Lunghezza Stringa

Testo

Scrivi un programma che chiede all'utente di inserire una stringa e poi:

1. Usa un puntatore per contare il numero di caratteri nella stringa (**senza usare strlen()**).
2. Usa il puntatore per stampare la stringa in ordine inverso

Esercizio 05

Inversione di un array

Testo

Scrivi un programma che chiede all'utente di inserire un array di 10 interi positivi, sfruttando **solo** l’aritmetica dei puntatori.

Il programma deve stamparne quindi l’inverso, di nuovo usando **solo** l’aritmetica dei puntatori.

Esercizio 06

Doppio di un array

Testo

Si scriva un programma che consenta all’utente di inserire un array di 5 interi positivi. Il programma deve salvare in un secondo array di stessa dimensione il doppio di ogni elemento.

Si usi **solo** l’aritmetica dei puntatori

Esercizio 07

Pari in pari e Dispari in Dispari

Testo

Si scriva un programma sfruttando solo l’aritmetica dei puntatori che, in un array di 10 interi positiviacquisito da input, conti quanti elementi pari ci sono in posizione pari, e quanti dispari in posizione dispari (contare la posizione “0” come pari).

Esempio:

[5,1,3,4,14,16,3,80,191,11]

Pari in pari: 1 (il 14)

Dispari in dispari: 2 (3 e 11)

Esercizio 07bis

Pari in pari e Dispari in Dispari

Testo

Dopo il programma precedente, salvare i “pari in pari” ed i “dispari in dispari” in due nuovi array

Esempio:

[5,1,3,4,14,16,3,80,191,11]

Pari in pari: 1 (il 14) -> pari_in_pari = contiene il 14 in posizione 0

Dispari in dispari: 2 (3 e 11) -> dispari_in_dispari = contiene 3 e 11 in posizione 0 e 1

Thank You!

Mail: luca1.alessandrini@polimi.it

Website: https://alessandriniluca.github.io/