Corso di Laurea triennale in Ingegneria Chimica

in condivisione con

Corso di Laurea triennale in

Ingegneria Navale e Scienze dei Materiali

Elementi di Informatica

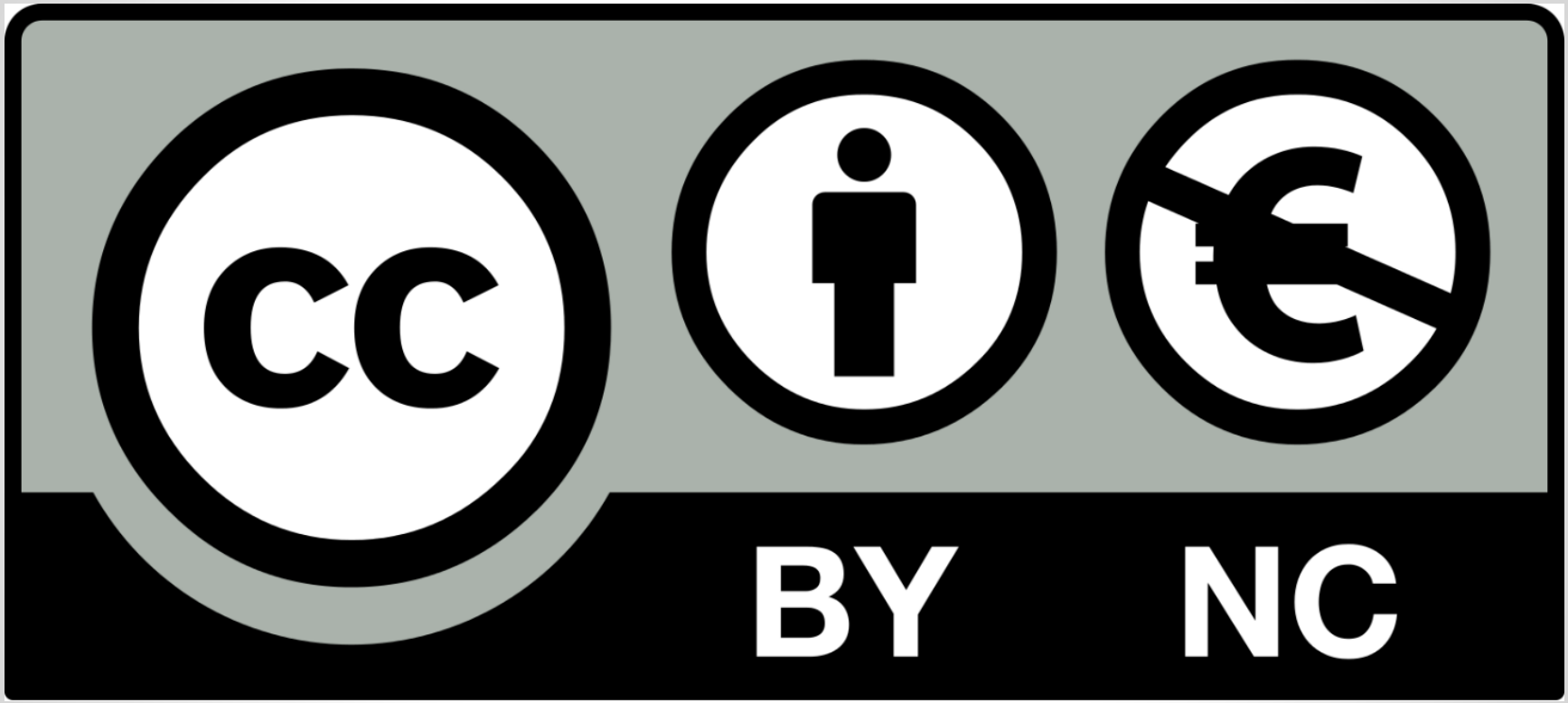
A.A. 2016/17

prof. Mario Barbareschi

Puntatori (cenni)

Informazioni di Licenza

- Questo lavoro è licenziato con la licenza Creative Commons BY-NC



- Per consultare una copia della licenza visita: http://creativecommons.org/licenses/by-nc/3.0/legalcode

I puntatori

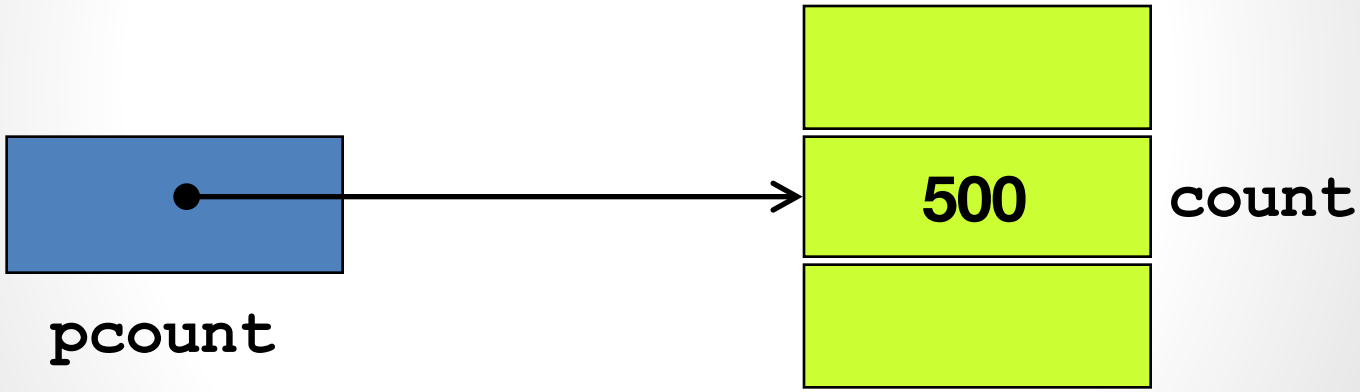
- Il C++ prevede puntatori a funzione e puntatori a dati di qualsiasi natura, semplici o strutturati.
- In particolare il puntatore viene utilizzato :
  - per il riferimento a liste di variabili dinamiche;
  - per il riferimento a funzioni;
  - per gli array, in particolare per l'elaborazione di stringhe;
  - per i parametri formali di funzione, in alternativa allo scambio per riferimento;
  - per il riferimento a strutture dati restituite da funzioni;
  - per il riferimento a locazioni specifiche della memoria, associate a dispositivi hardware (ad esempio per l'ingresso-uscita).

Il tipo Puntatore in C++

- Il puntatore del C++ è **strettamente tipizzato**;
- Un tipo che può assumere come insieme dei valori **tutti gli indirizzi di memoria**:
  - Dunque la sua dimensione è strettamente collegata all'architettura del processore (e.g. 32 bit oppure 64 bit).
- Un puntatore ad un variabile di tipo T identifica l'indirizzo di memoria della variabile.

Esempio di Puntatore in C++

- `typedef int *pint;`
- `pint pcount;`
- `int count=500;`
- `pcount=&count;`

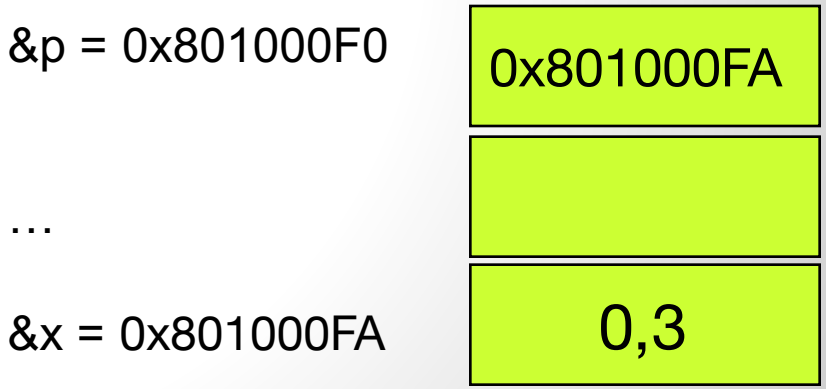


Uso dei puntatori

- `p` è un variabile di tipo puntatore
- `p=&x` al puntatore p viene assegnato l'indirizzo di x
- `*p` denota la variabile puntata da p (dereferenziazione)

Esempio:

Sia x una variabile di indirizzo 0x801000FA e valore 0,3;
p è un puntatore ad x;



Esempi di Assegnazione

```
int i = 1;
int j = 2;
int* ref1 = &i;
int* ref2 = &j;
```

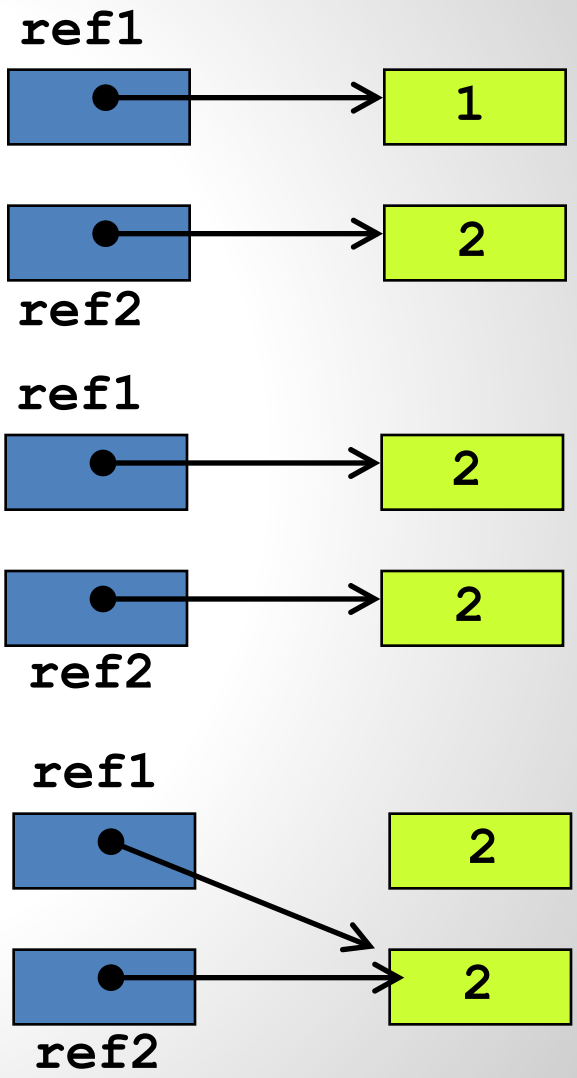
Inizializza ref1 e ref2 con l'indirizzo di memoria delle variabili i e j, rispettivamente

```
*ref1 = *ref2;
```

Copia il valore puntato da ref2 all'interno della locazione di memoria puntata da ref1

```
ref1 = ref2;
```

Copia l'indirizzo di memoria di ref2 in ref1



Operazioni con i Puntatori 1/3

- char* pc; //pc è un puntatore a variabile di tipo carattere
- int* pi; //pi è un puntatore a variabile di tipo intero
- float* pf;//pf è un puntatore a variabile di tipo float
- double* pd; //pd è un puntatore a variabile di tipo doppia precisione

Operazioni con i Puntatori 2/3

- Il linguaggio C++ tratta esplicitamente le espressioni su dati di tipo puntatore (**aritmetica dei puntatori**).
- Sono previsti i seguenti operatori:
- l'operatore di assegnazione tra puntatori che puntano allo stesso tipo T*;
- gli operatori unari di incremento (++) e decremento (--) unitari (in forma prefissa e postfissa ++p, p++, --p, p--);
- l'operatore di somma o differenza tra un puntatore ed un intero (p+i punta all'elemento che segue di i posizioni quello puntato da p);

Operazioni con i Puntatori 3/3

```
T * p;
```

```
p = p+1;
```

```
p = p+sizeof(T);
```

- l'incremento a p è tale che p+1 punta all'area di memoria immediatamente successiva a quella impegnata dall'elemento puntato da p.

- Questa tecnica è particolarmente utile per i puntatori ad array.

Puntatore a Dati Strutturati

- Un puntatore può puntare a variabili strutturate, in particolare esamineremo:
  - puntatori ad array;
  - puntatori a record;

- Il puntatore ad array può utilmente essere sostituito dalla notazione a[i] tipica degli array

I Puntatori ad Array 1/2

- Un puntatore ad un array a è una variabile che punta alla prima locazione dell'array (a[0] in C++);
- considerando che i successivi elementi dell'array sono allocati in posizioni contigue, esso consente di puntare anche a tutti gli altri elementi dell'array.

I Puntatori ad Array 2/2

```
T  a [dim];   // a è un array di dim elementi
              // di tipo T
T* p;    // p è un puntatore a T
```

```
p=a;
```

```
p=&a[0];
```

- **Un puntatore ad array è dunque di per sé un puntatore al tipo T degli elementi dell'array.**

Puntatori ad Array: esempio

```
const int dim=100;
float  a [dim];    /* a è un array di dim
                      elementi di tipo float*/
float* p;        // p è un puntatore a float

/*azzeramento di tutti gli elementi di un array di 100
elementi*/
for (p = a, int i=0; i<dim; i++, p++)
     *p= 0;
```

I Puntatori a Record 1/2

- Un puntatore ad un record è una variabile che punta all'indirizzo di memoria ove il record è allocato;
- Esso è molto utile nella realizzazione dei tipi dinamici.

```
// Dichiarazioni di un tipo strutturato Ts
struct  Ts { // Ts è un tipo strutturato
          D1;
          .....;
          Dn ;  } ;

Ts r; // r è variabile di tipo Ts

// Dichiarazione di un tipo puntatore a Ts
typedef  Ts* Punt;
// Dichiarazione  di variabile di tipo puntatore a Ts
Punt p;
```

I Puntatori a Record 2/2

```
Punt p = &r;  // p è una  variabile puntatore
   inizializzata ad r
```

- p può essere ridefinito nel corso del programma

```
p = &r1;//ora p punta alla variabile r1 di tipo
   Ts
```

- Accesso alla singola componente:

```
(*p).Di;
```

```
p->Di;
```

Puntatori a Record: esempio

```
// Definizione di tipo struttura:
struct Abbonati { // definisce la struttura Abbonati
          String nominativo;
          String indirizzo;
          int numTel; };

typedef Abbonati* Pabb;

// Definizione di variabili
Abbonati a; // a una variabile di tipo Abbonati
Pabb p= &a; // p, di tipo Pabb, è inizializzato ad a

// Accesso alla componente nominativo mediante puntatore
p->nominativo;
```

Parametro di tipo puntatore – 1/2

- Il generico argomento p di tipo puntatore è solitamente scambiato per valore, nel senso che l'indirizzo viene ricopiato nel corrispondente parametro formale.
- Tale parametro può essere di ingresso, di uscita, o di ingresso-uscita.
- L'uso del puntatore come parametro di scambio costituisce una soluzione alternativa all'impiego dello scambio per riferimento
- Lo scambio per riferimento è però più potente in quanto il compilatore tratta automaticamente il parametro formale con la tecnica dell'indirizzamento indiretto.

Parametro di tipo puntatore – 2/2

```
//swap con puntatori
void swap(char* px,char* py){
   char t;
   t=*px;
   *px=*py;
   *py=t;
}

//programma chiamante
char x, y;
x='a'; y='b';
swap(&x, &y);
```

```
//swap con riferimenti
void swap(char& x,char& y){
   char t;
   t=x;
   x=y;
   y=t;
}

//programma chiamante
char x, y;
x='a'; y='b';
swap(x, y);
```

Valore restituito di tipo puntatore – 1/2

- Una funzione può restituire un valore di tipo puntatore (cioè un indirizzo):

```
T* f();  //la funzione restituisce un
         » //valore di tipo puntatore a T
```

- ove T è il tipo puntato.

- Il puntatore restituito punta a un'**area associata a uno dei parametri** di ingresso della funzione.

Valore restituito di tipo puntatore – 2/2

```
//ricerca il carattere c nella stringa s e
//restituisce il puntatore alla prima occorrenza
char* strchr(const char s[], const char c)

char s[]="abcdef";
char* p=strchr(s,'d');   //p punta a s[3]
```

Argomento e valore restituito di tipo puntatore

- È possibile che una funzione abbia un **argomento** di tipo T* e un **valore restituito** del medesimo tipo:
- `T* f(...T*...);`

- In tal caso sono ammesse istruzioni del tipo:
- `f(...f(...p...)...)`
- Che producono l'esecuzione concatenata della funzione f.

- Esempio:
- `//concatenazione di stringhe`
- `strcat(strcat(a,b),c);`

Esercizio 1:

- Realizzare un programma che:
  - Crea una variabile vettore v di MAX_DIM posizioni
  - Chiede all'utente un numero < MAX_DIM
  - Chiama una funzione che popola il vettore con i quadrati fino a MAX_DIM
    - Passaggio per puntatore
  - Chiama una funzione che stampa il vettore
    - Passaggio per puntatore
  - Chiama una funzione che restituisce il puntatore alla prima occorrenza di un quadrato scelto dall'utente
    - Restituisce un puntatore
  - Stampa il vettore da quel puntatore in poi