Corso di Programmazione I

I puntatori in C e C++

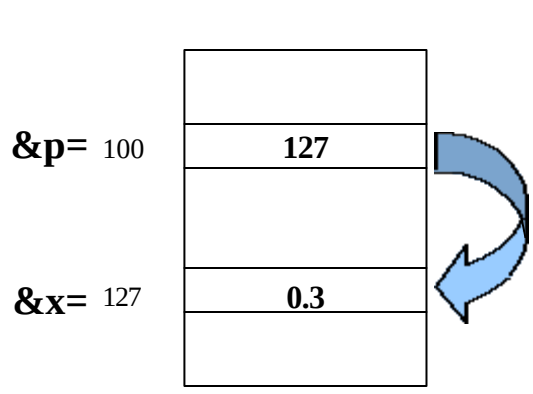
I puntatori in C

- 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 locazioni specifiche della memoria, associate a dispositivi hardware (ad esempio per l’ingresso-uscita)

I puntatori in C: notazioni

- **p** è una variabile di tipo puntatore
- **p=&x** al puntatore p viene assegnato l'indirizzo di x
- ***p** denota la variabile puntata da p

Esempio:
sia x una variabile di indirizzo 127 e valore 0,3;
p è un puntatore ad x;



I puntatori in C: operazioni (1/2)

```
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 precis.
```

- Il linguaggio C tratta esplicitamente le espressioni di tipo puntatore;
- in particolare 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
      - ++p, --p;
    - in forma postfissa
      - p++, p--;
  - l'operatore di somma o differenza tra un puntatore ed un intero
    - p. es.: p+i punta all'elemento che segue di i posizioni quello puntato da p;

Esercizio

1. Realizzare un programma che definisca e inizializzi due variabili di tipo intero (**`i1`** e **`i2`**) e due variabili di tipo float (**`f1`** e **`f2`**).
2. Stampare i valori delle variabili **`i1`**, **`i2`**, **`f1`** e **`f2`**.
3. Definire ed inizializzare un puntatore per ciascuna di queste variabili (siano **`pi1`**, **`pi2`**, **`pf1`**, **`pf2`**), associando i puntatori alla corrispondente variabile.
4. Stampare attraverso i puntatori i valori delle variabili **`i1`**, **`i2`**, **`f1`** e **`f2`**.
5. Cambiare attraverso i puntatori i valori delle variabili **`i1`**, **`i2`**, **`f1`** e **`f2`**.
6. Stampare (senza usare i puntatori) i valori delle variabili **`i1`**, **`i2`**, **`f1`** e **`f2`**.
7. Stampare i valori delle variabili puntatore **`pi1`**, **`pi2`**, **`pf1`**, **`pf2`**.
8. Stampare mediante l'operatore sizeof la dimensione delle variabili **`i1`**, **`i2`**, **`f1`** e **`f2`**.
9. Confrontare i risultati ottenuti e dedurre come avviene la allocazione delle variabili nella memoria.
10. Se le variabili sono state definite come variabili locali (globali) al main, definirle come variabili globali (locali) e controllare come cambia l'allocazione.

I puntatori in C: operazioni (2/2)

```
T* p;
```

`p=p+1;` → `p=p+sizeof(T);`

operazione logica

operazione algebrica

- 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

Esercizio

- Scrivere un programma che definisca variabili di vari tipi (**v1**,…,**vn**)
- Per ciascuna di queste, mediante l'operatore sizeof, stampare a video la dimensione di ciascun tipo
- Per ciascuna variabile creare un corrispondente puntatore (**p1**,…,**pn**), assegnarne il valore e stamparlo
- Stampare il valore di ciascun puntatore creato dopo averlo incrementato di 1 e di 2 (**p1+1**, **p1+2**, **p2+1**, **p2+2**, …, **pn+1**, **pn+2**)

Puntatori a dati strutturati

- Un puntatore può puntare a variabili appartenenti ad un tipo strutturato.
- Esamineremo i seguenti casi:
  - puntatore ad array;
  - puntatore a record.

Puntatori ad array (1/3)

- La variabile identificativa di un array **a** è un puntatore alla prima locazione dell'array (**a[0]** in C);
- considerando che i successivi elementi dell'array sono allocati in posizioni contigue, essa consente di puntare anche a tutti gli altri elementi dell'array.

```
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];`

- La variabile identificativa di un array è dunque di per sé un puntatore al tipo **T** degli elementi dell'array
- Il suo valore corrisponde con l'indirizzo del primo elemento dell'array

Puntatori ad array (2/3)

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
int i;

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

Si osservi che se **p** e **q** sono due puntatori allo stesso array **a**, **p** punta all'elemento **a[i]** e **q** all'elemento **a[j]**, la differenza **q-p** fornisce la "distanza" tra i due elementi, cioè **j–i**.
Per questo:

```
for (p=a; p-a<dim; p++) //forma alternativa per il for
  *p = 0;
```

Puntatori ad array (3/3)

- Tra i casi elencati, il puntatore ad array può utilmente essere sostituito dalla notazione p[i] tipica degli array.

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

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

p[i] ⟺ *(p + i)

Esercizio

- Scrivere un programma che definisca un array di 10 interi.
- Si stampi il valore degli elementi.
- Si trovino alcune possibili vie alternative per azzerare tutti gli elementi dell'array.
- Ancora utilizzando diversi approcci, si ristampi l'array per verificare che l'operazione di azzeramento sia effettivamente avvenuta.

Esercizio

- Si definisca un tipo matrice costituito da 4 righe e 5 colonne.
- Si stampi l'indirizzo di ciascun elemento della matrice.
- Si deduca come gli elementi di una matrice vengano organizzati dal compilatore nella memoria del calcolatore.

Puntatori a record (1/3)

- 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 delle strutture dati dinamiche

```
//  Dichiarazioni di un tipo strutturato Ts
typedef struct {  //Ts è un tipo strutturato
  Tipo1 Campo1;
  .....;
  TipoN CampoN ;
} Ts;

Ts r;   //r è variabile di tipo Ts

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

Puntatori a record (2/3)

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

p può essere riassegnata nel corso del programma

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

Accesso alla singola componente:

`(*p).Campo1;` ⟺ `p->Campo1;`

Puntatori a record (3/3)

Esempio

```
//Definizione di tipo struttura:
typedef struct {
  Stringa nominativo;
  Stringa indirizzo;
  int numTel ;
} Abbonato;   //  definisce struttura di nome Abbonato

typedef Abbonato* Pabb;

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

//Accesso al campo nominativo mediante puntatore
p->nominativo;
```

Esercizio

- Definire un tipo struttura **Ts** contenente un certo numero di campi di diverso tipo
- Istanziare una variabile **s** di tipo **Ts**
- Definire un puntatore ps alla struttura e inizializzarlo all'indirizzo della variabile **s**
- Attraverso il puntatore assegnare un valore a ciascun campo della struttura **s**
- Stampare tutti i valori assegnati
- Stampare l'indirizzo della variabile **s**
- Stampare l'indirizzo di ciascun campo della variabile **s**