Università degli Studi di Cassino e del Lazio Meridionale

Corso di Fondamenti di Informatica

Puntatori

Anno Accademico 2016/2017

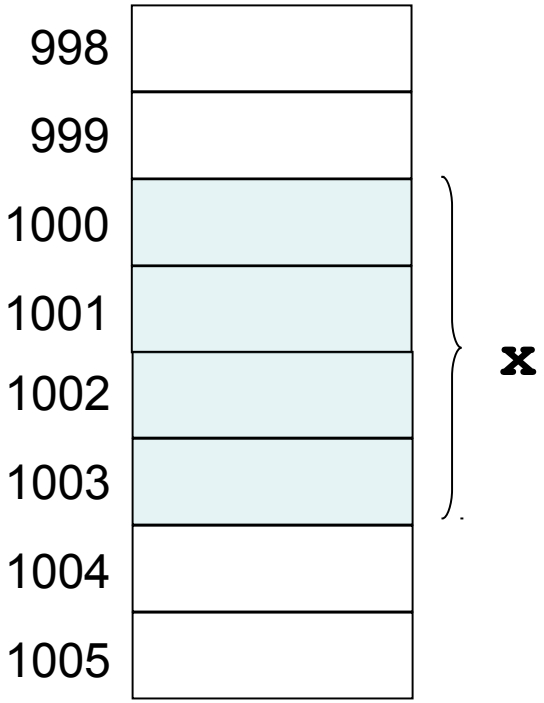
Francesco Tortorella

Variabili, registri ed indirizzi

- La definizione di una variabile implica l'allocazione (da parte del compilatore) di registri di memoria. Il numero di registri allocati dipende dal tipo della variabile.
- Alla porzione di memoria allocata si accede tramite l'identificatore della variabile. Questo ci risparmia di preoccuparci in quale particolare locazione la variabile sia realmente allocata.
- È il compilatore a creare e gestire la corrispondenza tra identificatore della variabile e indirizzo della locazione in memoria.

Variabili, registri ed indirizzi

- Esempio:
  `int x;`
- Con l'istruzione viene definita una variabile intera **`x`** che occupa 4 registri da 1 byte a partire dall'indirizzo 1000.



Variabili, registri ed indirizzi

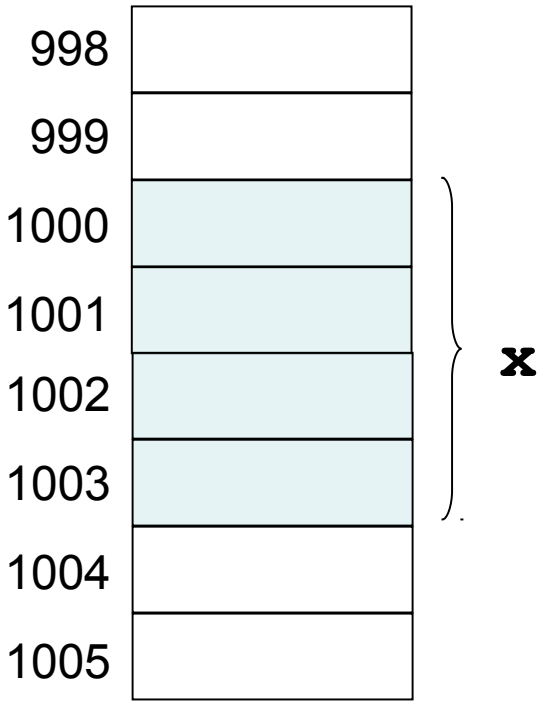
- Il C++ dà la possibilità di accedere esplicitamente all'indirizzo di una variabile tramite l'operatore & (operatore di riferimento o di *reference*) prefisso all'identificatore della variabile.

  `int x;` variabile **`x`**

  `&x` indirizzo della variabile **`x`**

Variabili, registri ed indirizzi

- Esempio:
  `int x;`
- In questo caso `&x` sarà uguale a 1000.
- **ACHTUNG !**
  L'operatore & si può applicare solo alle variabili (o, più precisamente, a ***l-value***)



Variabili, registri ed indirizzi

```
#include <iostream>
using namespace std;

int main(){
    int x=3;

    cout << "Valore di x: " << x << endl;
    cout << "Indirizzo di x: " << &x << endl;
    return (0);
}
```

```
Valore di x: 3

Indirizzo di x: 0x22ff48
```

indirizzo esadecimale

Puntatori

- Il C++ permette di definire delle variabili di tipo “puntatore” cui si possono assegnare gli indirizzi di variabili di un particolare tipo.
- La definizione di tali variabili (dette **puntatori**) richiede la specificazione del tipo “puntato”, seguito da un ‘*’.
- Es.: definizione di un puntatore a `int`

```
int *p;
```

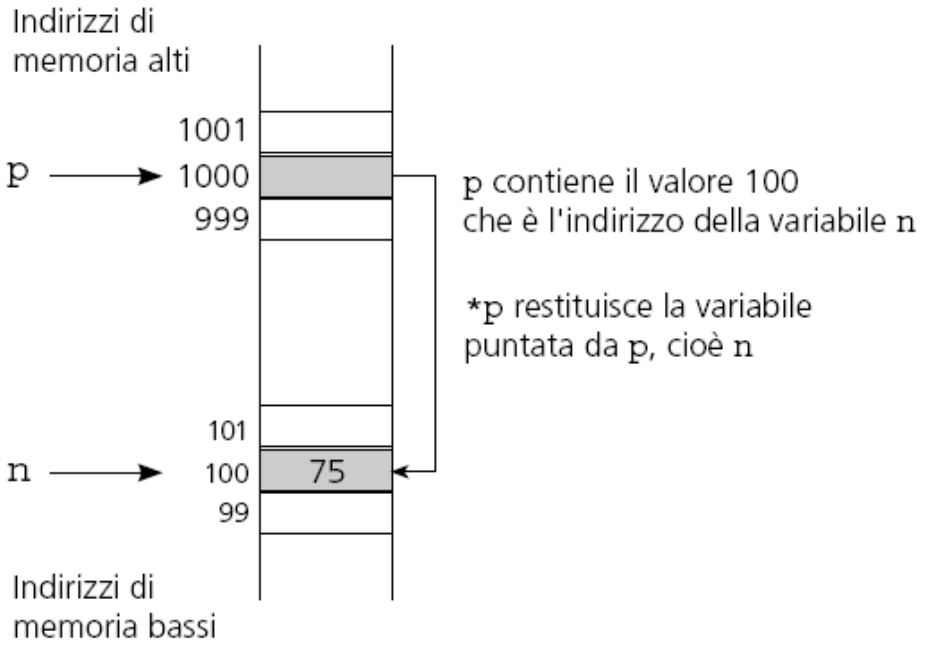
Puntatori

Di fatto una variabile di tipo *puntatore al tipo T* contiene l'indirizzo di memoria di una variabile di *tipo T.*

Esempio:

```
int n;
int *p;

n = 75;
p = &n;
```

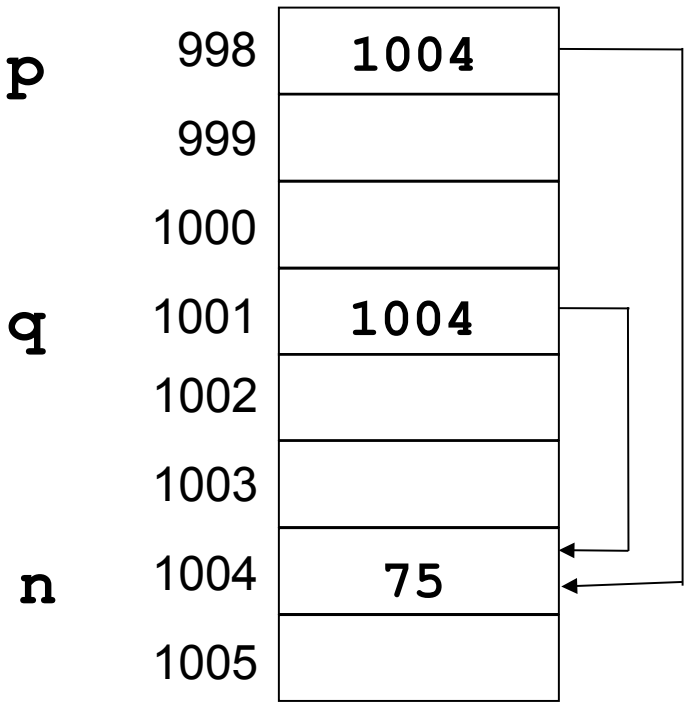


Puntatori

- È possibile fare assegnazioni tra puntatori.
- Esempio:

```
int n=75;
int *p,*q;
p = &n;
q = p;
```

- In questo modo due puntatori puntano alla stessa variabile.



Puntatori

- Tramite il puntatore è possibile accedere alla variabile puntata.
- Con l'operatore * (operatore di indirezione o di *dereference*) prefisso all'identificatore della variabile puntatore è possibile accedere direttamente alla variabile puntata, sia in lettura che in scrittura.
- In questo modo si crea un *alias* della variabile che può essere modificata tramite il puntatore.

Puntatori

```
#include <iostream>
using namespace std;

int main() {
    int x=3,y=5;
    int *p;

    p = &x;
    *p = 10;
    cout << "Valore di x: " << x << endl;

    p = &y;
    *p = 20;
    cout << "Valore di y: " << y << endl;

    return 0;
}
```

```
Valore di x: 10
Valore di y: 20
```

Puntatori

- Non è possibile fare assegnazioni tra puntatori di tipo diverso.

  Esempio:

```
int x=3, *p;
char c='A', *t=&c;
p=t;
```

  non è consentito

- Perché ? In fondo entrambi contengono un indirizzo.
- Domanda fondamentale: perché i puntatori si riferiscono a un tipo particolare ?

Puntatori

- Di fatto, `p` e `t` sono entrambi puntatori ed occupano lo stesso spazio in memoria (quanto ?).
- Bisogna però ricordare che c'è una profonda differenza tra i dati puntati :
  - Non sono dello stesso tipo
  - Non occupano lo stesso spazio in memoria
  - Non utilizzano la stessa rappresentazione dei dati
- Per accedere correttamente alla variabile puntata è quindi necessario che il puntatore "porti con sé" l'informazione sul tipo puntato.

Puntatore NULL

- Un puntatore non inizializzato può dare qualche problema perché, di fatto, punta ad una locazione casuale in memoria.
- È quindi necessario avere un valore “neutro” da poter assegnare ad un puntatore per segnalare che non indirizza alcun dato valido in memoria.
- Per questo scopo si usa la costante simbolica **`NULL`** (definita in vari header tra cui **`iostream`**).

Puntatore NULL

- Un puntatore può essere inizializzato a **`NULL`**:
  `int *p = NULL;`
- È possibile operare un confronto con **`NULL`**:
  `if(p != NULL)`…

Aritmetica dei puntatori

- È possibile compiere operazioni aritmetiche tra puntatori e interi.
- In particolare, sono ammesse:
  - Addizione di un puntatore ed un intero
  - Sottrazione di un intero da un puntatore
  - Differenza tra due puntatori

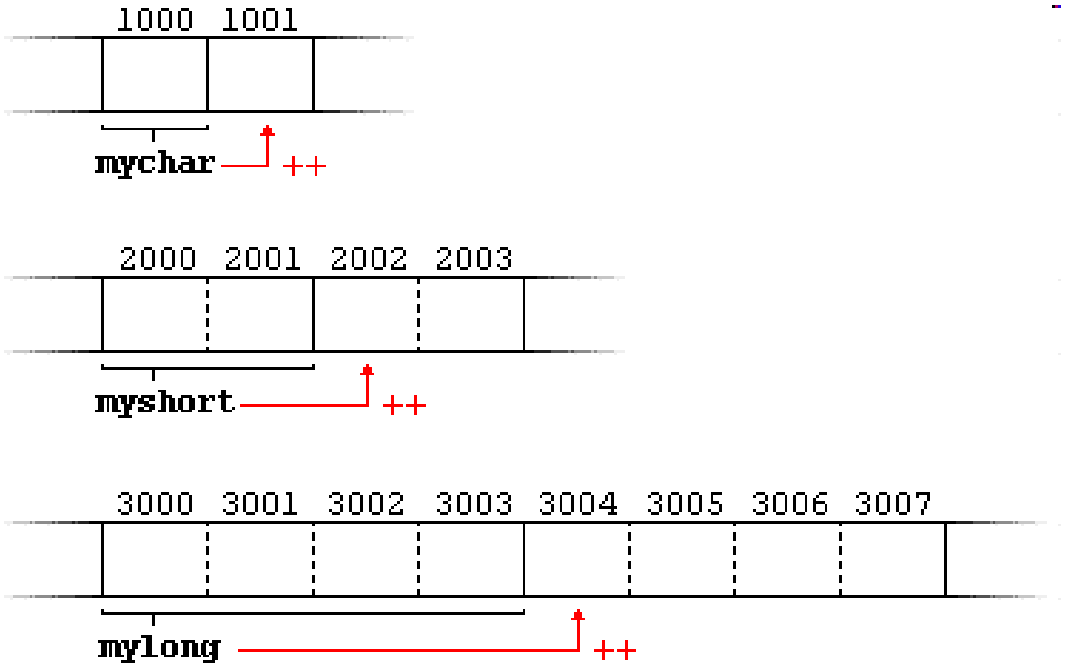
Aritmetica dei puntatori

- Che cosa significa sommare un puntatore ed un intero ? Qual è il risultato ?
- Il risultato è comunque un puntatore.
  A cosa ?
- Se un puntatore a un tipo T viene incrementato di 1, il suo valore viene di fatto incrementato di una quantità pari alla dimensione in byte del tipo T.

Aritmetica dei puntatori

```
char *mychar;
short *myshort;
long *mylong;
```

L'incremento individua l'indirizzo della variabile di tipo T immediatamente seguente in memoria



Aritmetica dei puntatori

- Lo stesso vale per il decremento.
- Che cosa succede se invece si somma (o si sottrae) un valore *k* > 1 ?
- **`p+k`**: il risultato è uguale al valore di p incrementato di k**d* dove *d* è la dimensione in byte del tipo puntato.
- Lo stesso vale per **`p-k`**

Aritmetica dei puntatori

- La differenza tra puntatori è possibile solo tra puntatori dello stesso tipo.
- Il risultato della differenza è uguale alla differenza tra i due indirizzi diviso la dimensione in byte del tipo puntato.
- In pratica, la differenza **`p-q`** tra due puntatori **`p`** e **`q`** è uguale all'intero **`w`** tale che **`p+w=q`**.

Puntatori a costanti

- È possibile definire puntatori a costanti, in modo che la variabile puntata non possa essere modificata tramite indirezione del puntatore.
- La dichiarazione prevede di anticipare la parola chiave const alla consueta dichiarazione. Es.:

```
const int *px;
int x=3,y;

px=&x;
y=*px;
*px=5;
```

possibile

illegale