Carlo Gaibisso
Bruno Martino

Programmazione e Laboratorio di Programmazione

Manualistica 2
Gli operatori

Carlo Gaibisso
Bruno Martino

Operatore &

- **Sintassi :**

  &nome_variabile

- **Valore:**

  indirizzo della variabile nome_variabile

- **Esempio:**

```
int *x;
int y;

x = &y;
```

Carlo Gaibisso
Bruno Martino

Operatore *

- **Sintassi :**

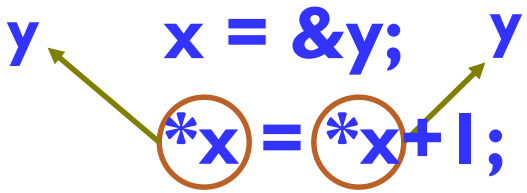
  *nome_puntatore

- **"Valore":**

  il nome della variabile il cui indirizzo è memorizzato in nome_puntatore

- **Esempio:**

```
int *x, y;
y = 1;
x = &y;
*x = *x+1;
```



Carlo Gaibisso
Bruno Martino

Operatore di assegnamento

- **Sintassi:**

  nome_variabile = espressione

  espressione

- **Valore:**

  è il valore di espressione

- **Modifiche allo stato della memoria:**

  assegna alla variabile nome_variabile
  il valore di espressione

Carlo Gaibisso
Bruno Martino

Operatore di assegnamento

- **Attenzione:**

  il tipo di nome_variabile e quello di espressione "devono" coincidere

Carlo Gaibisso
Bruno Martino

Operatore di assegnamento

- **Esempio:**

```
// sorgente: Lezione_XI\OpAss_1.c
// programma che mostra il comportamento
// dell'operatore di assegnamento
#include <stdio.h>
int main ()
        {
        // definizione e inizializzazione delle variabili
        int x, y;
        y=2;
        // visualizza il valore di y assegnandolo
        // al tempo stesso a x
        printf("\nY: %d", x=y);
        // visualizza il valore di x
        printf("\nX: %d", x);
        return(1);
        };
```

Carlo Gaibisso
Bruno Martino

Operatore di assegnamento

- **Compilazione:**

```
D:\Codice\Lezioni\Lezione_XI>gcc -Wall OpAss_1.c
```

- **Esecuzione:**

```
D:\Codice\Lezioni\Lezione_XI>a

Y: 2
X: 2
```

Carlo Gaibisso
Bruno Martino

Operatore di assegnamento

- **Sintassi:**

  *nome_puntatore = espressione

  espressione

- **Valore:**

  è il valore di espressione

- **Modifiche allo stato della memoria:**

  assegna alla variabile il cui indirizzo è memorizzato in nome_puntatore il valore di espressione

Carlo Gaibisso
Bruno Martino

Operatore di assegnamento

- **Esempio:**

```
// sorgente: Lezione_XI\OpAss_2.c
// programma che mostra il comportamento
// dell'operatore di assegnamento
#include <stdio.h>
int main ()
        {
        // definizione e inizializzazione delle variabili
        int x, y, *punx, *puny;
        punx=&x; puny=&y;
        *puny=2;
        // visualizza il valore di y assegnandolo
        // al tempo stesso a x
        printf("\nY: %d", *punx=*puny);
        // visualizza il valore di x
        printf("\nX: %d", *punx);
        return(1);
        };
```

Operatore di assegnamento

- **Compilazione:**

```
D:\Codice\Lezioni\Lezione_XI>gcc -Wall OpAss_2.c
```

- **Esecuzione:**

```
D:\Codice\Lezioni\Lezione_XI>a

Y: 2
X: 2
```

Carlo Gaibisso
Bruno Martino

Operatore di auto-incremento prefisso

- **Sintassi:**

  ++nome_variabile

- **Valore:**

  il valore di nome_variabile incrementato di 1

- **Modifiche allo stato della memoria:**

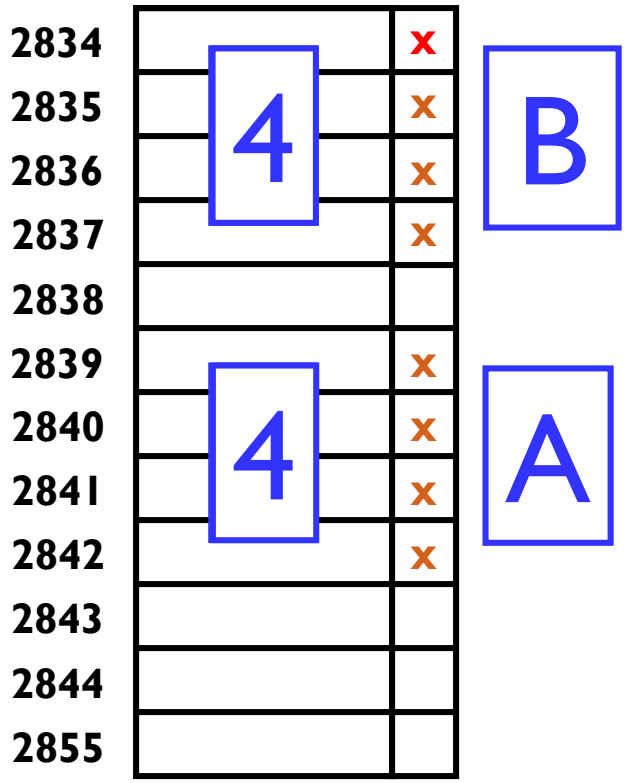
  incrementa di 1 il valore di nome_variabile

Carlo Gaibisso
Bruno Martino

Operatore di auto-incremento prefisso

- **Esempio:**

B=++A;



Carlo Gaibisso
Bruno Martino

Operatore di auto-incremento postfisso

- **Sintassi:**

  nome_variabile++

- **Valore:**

  il valore di nome_variabile

- **Modifiche allo stato della memoria:**

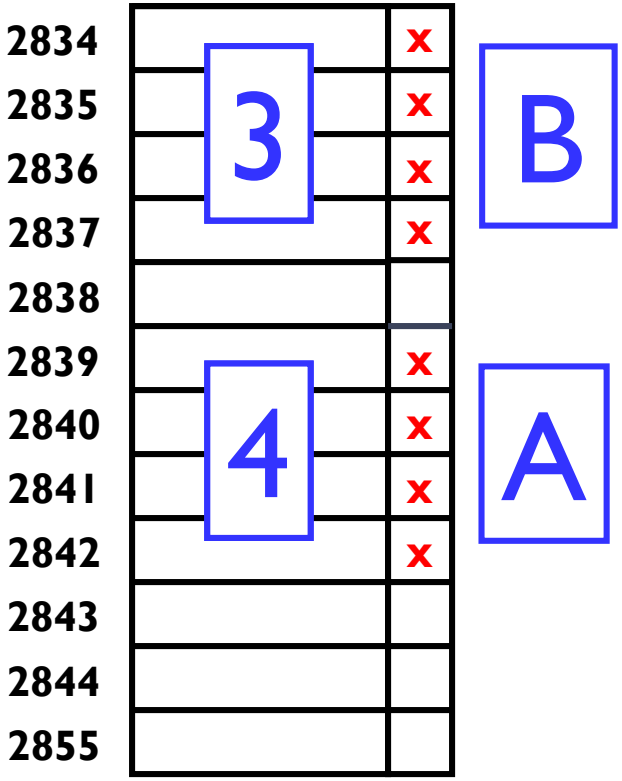
  incrementa di 1 il valore di nome_variabile

Carlo Gaibisso
Bruno Martino

Operatore di auto-incremento postfisso

- **Esempio:**

B=A++;



Operatore di auto-decremento prefisso

- **Sintassi:**

  `--nome_variabile`

- **Valore:**

  il valore di `nome_variabile` decrementato di `1`

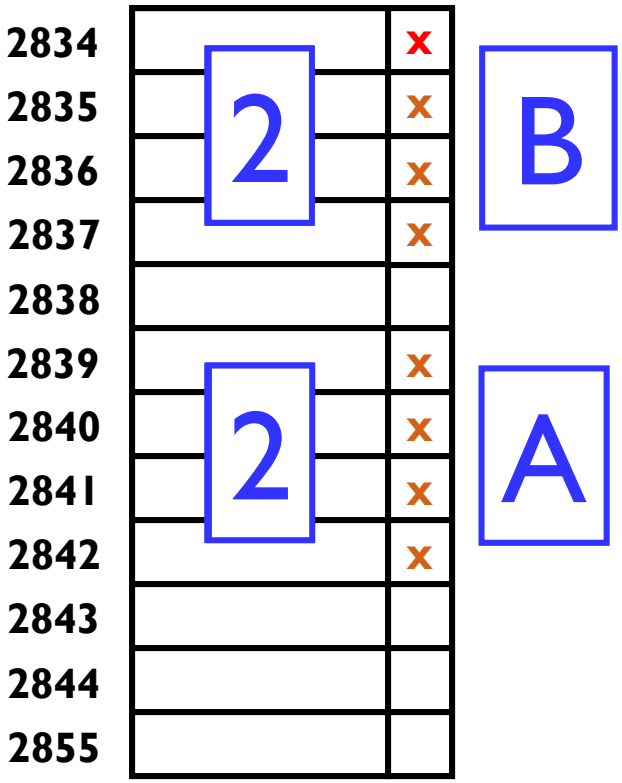
- **Modifiche allo stato della memoria:**

  decrementa di `1` il valore di `nome_variabile`

Operatore di auto-decremento prefisso

- **Esempio:**

B=--A;



Carlo Gaibisso
Bruno Martino

Operatore di auto-decremento postfisso

- **Sintassi:**

  nome_variabile--

- **Valore:**

  il valore di nome_variabile

- **Modifiche allo stato della memoria:**

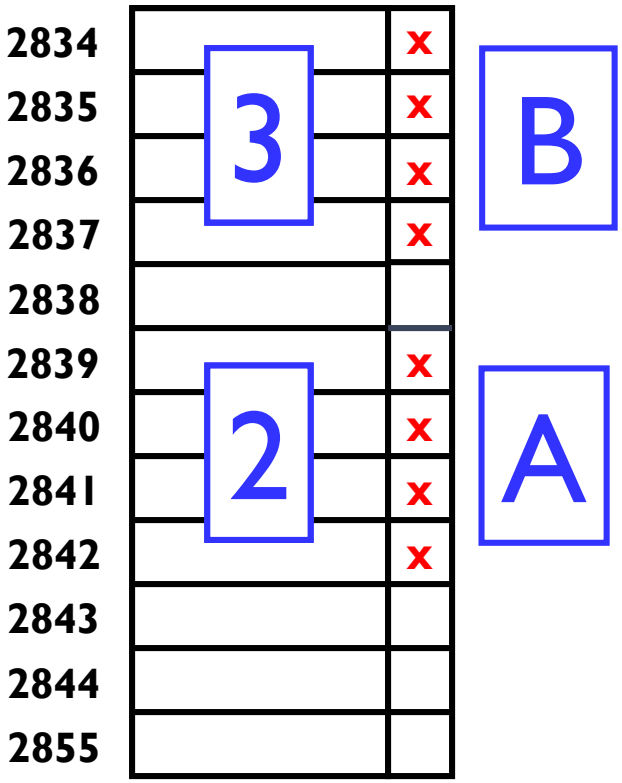
  decrementa di 1 il valore di nome_variabile

Carlo Gaibisso
Bruno Martino

Operatore di auto-decremento postfisso

- **Esempio:**

B=A--;



Operatori di auto-incremento e decremento

- **Esempio**

```
// sorgente Lezione_XI\AutoIncrDecr.c

// Mostra il comportamento degli operatori
// di autoincremento e autodecremento

#include <stdio.h>

int main ()
        {
        int y = 1;

    printf("\nValore di y +suffisso = %d\n", y++);
    printf("Valore di y +prefisso = %d\n", ++y);
    printf("Valore di y -prefisso = %d\n", --y);
    printf("Valore di y -suffisso = %d\n", y--);
    printf("Valore finale di y = %d\n", y);

        return(1);
        }
```

1
3
2
2
1

Operatori di auto-incremento e decremento

- **Compilazione:**

```
D:\Codice\Lezioni\Lezione_XI>gcc AutoIncrDecr.c
```

- **Esecuzione:**

```
D:\Codice\Lezioni\Lezione_XI>a

Valore di y +suffisso = 1
Valore di y +prefisso = 3
Valore di y -prefisso = 2
Valore di y -suffisso = 2
Valore finale di y = 1
```

Operatori di relazione

- **Sintassi:**
  a) espr_1 == espr_2 uguale
  b) espr_1 != espr_2 diverso
  c) espr_1 > espr_2 maggiore
  d) espr_1 >= espr_2 maggiore o uguale
  e) espr_1 < espr_2 minore
  f) espr_1 <= espr_2 minore o uguale
- **Valore:**
  1 se i valori di espr_1 e espr_2 si trovano nella relazione specificata
  0 altrimenti

Operatori di relazione

- **Esempio**

```
// sorgente: Lezione_XI\OpRel.c

// programma che mostra il comportamento degli
// operatori di relazione

#include <stdio.h>

int main ()
  {
  printf ("\nValore di 3 < 5: %d\n",(3 < 5));
  printf ("Valore di 3 > 5: %d\n",(3 > 5));
  printf ("Valore di 3 == 5: %d\n",(3 == 5));
  printf ("Valore di 3 != 5: %d\n",(3 != 5));
  printf ("Valore di 5 == 5: %d\n",(5 == 5));

  printf ("Numero di relazioni soddisfatte: %d", \
          (3 < 5)+(3 > 5)+(3 == 5)+(3 != 5)+(5 == 5));

  return(1);
  }
```

```
1
0
0
1
1
3
```

Carlo Gaibisso
Bruno Martino

Operatori di relazione

- **Compilazione:**

```
D:\Codice\Lezioni\Lezione_XI>gcc -Wall OpRel.c
```

- **Esecuzione:**

```
D:\Codice\Lezioni\Lezione_XI>a

Valore di 3 < 5: 1
Valore di 3 > 5: 0
Valore di 3 == 5: 0
Valore di 3 != 5: 1
Valore di 5 == 5: 1
Numero di relazioni soddisfatte: 3
```

Carlo Gaibisso
Bruno Martino

Operatori "logici"

- **Sintassi:**

  $espr_1$ || $espr_2$ "or"

- **Valore:**

  0 se $espr_1$ e $espr_2$ valgono entrambe 0

  1 altrimenti

- **Sintassi:**

  $espr_1$ && $espr_2$ "and"

- **Valore:**

  1 se $espr_1$ e $espr_2$ sono entrambe $\neq 0$

  0 altrimenti

Carlo Gaibisso
Bruno Martino

Operatori "logici"

- **Sintassi:**

  !espr **"not"**

- **Valore:**

  1 **se espr vale 0**

  0 **altrimenti**

Carlo Gaibisso
Bruno Martino

Operatori "logici"

- **Esempio**

```
// sorgente: Lezione_XI\OpLog.c

// programma che illustra il comportamento degli
// operatori logici

#include <stdio.h>

int main ()
  {
printf ("\nValore di !(3 > 5): %d\n", !(3 > 5));

printf ("Valore di ((3 < 5) || (3 > 5)): %d\n", \
       ((3 < 5) || (3 > 5)));

printf ("Valore di ((3 < 5) && (3 > 5)): %d\n", \
       ((3 < 5) && (3 > 5)));

  return(1);
  }
```

Operatori "logici"

- **Compilazione:**

```
D:\Codice\Lezioni\Lezione_XI>gcc -Wall OpLog.c
```

- **Esecuzione:**

```
D:\Codice\Lezioni\Lezione_XI>a

Valore di !(3 > 5): 1
Valore di ((3 < 5) || (3 > 5)): 1
Valore di ((3 < 5) && (3 > 5)): 0
```