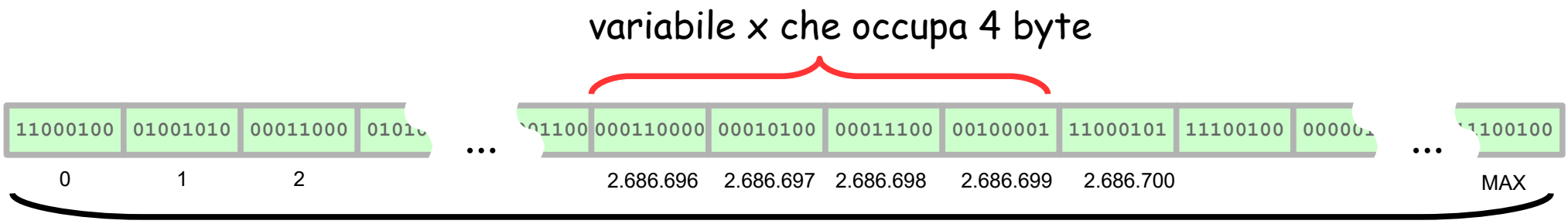
Puntatori

Indirizzi di memoria



spazio degli *indirizzi logici* associato al programma

L'**operatore &** consente di ottenere l'indirizzo di memoria della variabile a cui viene applicato.

```
int main() {
    int x;
    printf("%u", &x ); // stampa 2686696
}
```

Cos'è un puntatore

un **puntatore** è una variabile che contiene l'indirizzo di memoria di un'altra variabile.

Se un puntatore ***p*** contiene l'indirizzo della variabile ***x***,
si dice che " ***p punta a x*** ".

```
int main() {
    int x;     // dichiarazione una variabile di tipo "int"
    int* p;    // dichiarazione una variabile di tipo "puntatore a int"
    p = &x;    // ora p punta a x
}
```

Dichiarazione e inizializzazione

Si può anche **inizializzare un puntatore al momento della dichiarazione**

```
int main() {
    int x;
    int *p = &x;    // dichiarazione e inizializzazione di p
}
```

l'asterisco può essere "attaccato" al tipo di dato o al nome della variabile (indifferente) ... MA ATTENZIONE:

```
int main() {
    int* x,y,z;           // solo x è un puntatore! y e x sono di tipo "int".
    int *a, *b, *c;       // tre  variabili di tipo "puntatore a int"
    float *p1,*p2,*p3;    // tre variabili di tipo "puntatore a float"
}
```

Puntatori in C (primo esempio)

```
int main() {
  int x = 2;
  int *p;                    // dichiara p, di tipo "puntatore a int"
  printf("%d \n", x );       // stampa il contenuto (valore) di x
  printf("%d \n", &x );      // stampa l'indirizzo della variabile x
  p = &x;                    // metto in p l'indirizzo di x.
                             // ora p PUNTA a x
  printf("%d \n", p );       // stampa il contenuto di p (cioè l'indirizzo di x)
  printf("%d \n", &p );      // stampa indirizzo della variabile p
  printf("%d \n", *p );      // stampa il contenuto della variabile puntata
                             // da p (cioè il valore della variabile di x)
  *p = 100;                  // inserisce il valore 100 nell'area di memoria
                             // puntata da p (quindi equivale a scrivere
                             // x=100 )
   printf("%d \n",x);        // stampa il contenuto attuale di x (cioè 100)
}
```

Aritmetica dei puntatori

*Incrementare di n "unità" un puntatore equivale a incrementare il valore contenuto nel puntatore di n*sizeof(tipo)*

```
int x;
int *p = &x;
printf("%u",p);    // stampa 2686696
p = p + 1;
printf("%u",p);   // stampa 2686700
```

Puntatori e array

```
int arr[MAX];
```

Il nome di una variabile di tipo array rappresenta l'indirizzo del primo elemento dell'array

```
arr == &(arr[0])
```

*Vale la seguente equivalenza fra gli opreatori [] e **

```
arr[i] == *(arr+i)
```