9.3 Aritmetica dei puntatori

Un puntatore contiene un indirizzo e le operazioni che possono essere compiute su un puntatore sono perciò quelle che hanno senso per un indirizzo. Le uniche operazioni ammissibili sono dunque: l'incremento, per andare da un indirizzo più basso a uno più alto, e il decremento, per andare da un indirizzo più alto a uno più basso. Gli indirizzi sono per la memoria quello che sono gli indici per un array.
Gli operatori ammessi per una variabile di tipo puntatore sono:

```
+  ++  -  --
```

Ma qual è l'esatto significato dell'incremento o decremento di un puntatore? Non pensi il lettore che il valore numerico del puntatore corrispondente a un indirizzo venga incrementato come una qualunque altra costante numerica. Per esempio, se `pc` vale 10, dove `pc` è stato dichiarato:

```
char *pc;
```

non è detto che `pc++` valga 11 !
Nell'aritmetica dei puntatori quello che conta è il tipo base. Incrementare di 1 un puntatore significa far saltare il puntatore alla prossima locazione corrispondente a un elemento di memoria il cui tipo coincide con quello base. Per esempio, in:

```
int a[10];
char b[10];
int *pi;
char *pc;

pi = a;
pc = b;

pi = pi + 3;
pc = pc + 3;
```

le ultime due istruzioni che incrementano di 3 i puntatori `pi` e `pc` debbono essere interpretate in modo diverso. La prima,

```
pi = pi + 3;
```

significa spostare in avanti `pi` di tre posizioni, dove ogni posizione occupa lo spazio di un `int`. La seconda,

```
pc = pc + 3;
```

significa spostare in avanti `pc` di tre posizioni, dove ogni posizione occupa lo spazio di un `char`.
Più in generale si ha che, quando un operatore aritmetico è applicato a un puntatore `p` di un certo tipo e `p` punta a un elemento di un array di oggetti di quel tipo, `p+1` significa "prossimo elemento del vettore" mentre `p-1` significa "elemento precedente".
La sottrazione tra puntatori è definita solamente quando entrambi i puntatori puntano a elementi dello stesso array. La sottrazione di un puntatore da un altro produce un numero intero corrispondente al numero di posizioni tra i due elementi dell'array. Si osservi invece come sommando o sottraendo un intero da un puntatore si ottenga ancora un puntatore. Si considerino i tre esempi seguenti.

```
int v1[10];
int v2[10];
int i;
int *p;


i = &v1[5] - &v1[3];  /* 1 ESEMPIO */
printf("%d\n", i);    /* i vale 2 */

i = &v1[5] - &v2[3];  /* 2 ESEMPIO */
printf("%d\n", i);    /* il risultato è indefinito */

                      /* 3 ESEMPIO */
p = v2 - 2;           /* dove va a puntare p ? */
```

1. 1. *sottrazione tra indirizzi dello stesso vettore:*

   ```
   i = &v1[5] - &v1[3];
   ```

   corrispondente a un caso perfettamente legale;

2. 2. *sottrazione tra indirizzi di array diversi:*

   ```
   i = &v1[5] - &v2[3];
   ```

   corrispondente a un caso il cui risultato non è prevedibile;

3. 3. *sottrazione di una costante da un indirizzo ma nella direzione sbagliata:*

   ```
   p = v2 - 2;
   ```

   il puntatore `p` va a puntare due interi prima dell'inizio del vettore `v2` (per come sono avvenute le definizioni probabilmente si sconfina nello spazio riservato a `v1`, ma non è detto !).