- I puntatori ottenuti via new devono essere distrutti con delete (mai usare free());
- Mai liberare un'area di memoria (ad esempio con free()) già deallocata. Evitare errori logici nel codice che consentano l'insorgere di problematiche di questo tipo;
- Mai tentare di scrivere in un buffer residente in heap memory dopo la sua deallocazione. Evitare l'insorgere di errori logici di questo tipo.

7.1.8.8 Puntatori

- Gestire opportunamente i puntatori a NULL;

Esempio:

Forma non corretta:

```
char tmpchar1 (char *s)
{
  return *s;
}
//"s" == NULL → CRASH
```

Forma corretta:

```
char tmpchar1 (char *s)

{
   if (s == NULL) return '\0';
return *s;
}
```

7.1.8.9 Casting e problematiche di gestione delle variabili numeriche

- Il tipo NULL deve essere corretto mediante casting quando passato come parametro a una funzione;
- Ridurre al minimo le comparazioni fra interi di tipo signed. Se due interi di tipo signed vengono comparati, deve essere previsto il caso "minore di zero" (< 0), soprattutto quando la comparazione avviene con un valore costante.

Esempio:

Comparazione non signed:

```
if ((int)val1 < (unsigned int)val2)
/* in questo caso unsigned ha la precedenza essendo un tipo più grande di
signed. Entrambi i valori
(val1 e val2) vengono quindi
convertiti ad unsigned prima di essere comparati
*/
if ((int)val < sizeof(costante))
// l'operatore sizeof è unsigned
```

Comparazione signed:

```
if ((int)val < 256)
if (unsigned short)val1 < (short)val2)
/* la seguente comparazione dovrebbe, in base al tipo di compilatore, essere
signed perchè entrambi gli short dovrebbero essere convertiti a signed integer
prima di essere comparati
*/
```

- Evitare di utilizzare variabili signed integer come length specifier, ovvero come indicatori dell'allocazione/dimensione di un buffer o di un array.
- Evitare che un intero, a seguito di un'operazione di moltiplicazione, addizione o sottrazione, cresca oltre il suo valore massimo o decresca sotto il suo valore minimo. Ad esempio su architettura a 32 bit se un intero signed a 16 bit dal valore 32767 viene incrementato di una unità, il suo valore diverrà -32768, producendo un errore di overflow. È bene assicurarsi che questo genere di condizioni non si verifichi in alcun caso, soprattutto su input fornito dall'utente, in prossimità dell'allocazione di un buffer o della copia di dati da un buffer all'altro.