Gli argomenti delle macro devono essere accuratamente racchiusi in parentesi.

7.1.8.5 L'operatore sizeof e il passaggio di dati come parametri

Il passaggio della dimensione di una struttura dati come parametro a una funzione deve essere effettuato in maniera corretta, tramite l'utilizzo della funzione sizeof(). A tal proposito, è necessario sviluppare consapevolezza degli errori qui menzionati, e non ripeterli:

Esempio:
Forma non corretta:

```
strlen(struttura)
sizeof(ptr)
sizeof(*array)
/*
 * Dimensione di un solo elemento
 */
sizeof(array)
```

Forma corretta:

```
sizeof(struttura)
sizeof(*ptr)
sizeof(array)
/*
 * Dimensione di un solo elemento
 */
sizeof(array[0])
```

Gli argomenti delle macro devono essere accuratamente racchiusi in parentesi.

7.1.8.6 Allocazione dinamica

Il successo dei linguaggi C e C++ è dovuto alla grande flessibilità che offrono allo sviluppatore nella gestione diretta della memoria della macchina. Ciò offre illimitate possibilità, ma comporta anche rischi piuttosto elevati. Per mitigare tali rischi occorre adottare i seguenti suggerimenti:

- Lo spazio di memoria allocato dinamicamente (ad esempio con le funzioni malloc(), calloc() e realloc() deve essere appropriato alla dimensione dei dati che deve contenere;
- L'applicazione deve provvedere all'allocazione e alla deallocazione della memoria. Nell'ambito della programmazione multithreaded, vale lo stesso principio: ogni thread deve allocare e deallocare la propria memoria, senza delegare la deallocazione ad altri thread;
- Se si scrive codice C++ è meglio sfrutttare le caratteristiche peculiari di questo linguaggio, piuttosto che appoggiarsi alle strutture del C, mantenute per compatibilità. Esempio: utilizzare "new" invece che malloc(), calloc(), e realloc();

7.1.8.7 Deallocazione

- Gli array non devono essere cancellati come dati scalari;

Esempio:
Forma non corretta:

```
delete mioarray;
```

Forma corretta:

```
delete [ ] mioarray;
```

- Non devono esistere puntatori a risorse distrutte: contestualmente alla distruzione delle risorse vanno dereferenziati tutti i puntatori;
- I puntatori relativi alla memoria allocata dinamicamente devono essere impostati a NULL subito dopo essere stati rilasciati;
- I puntatori ottenuti via malloc(), calloc(), realloc() devono essere distrutti con free() (mai usare delete);