- Inizializzazioni
- Tutte le variabili locali devono essere inizializzate prima di essere utilizzate. Se sono inizializzate con valori "dummy" o momentanei devono essere reinizializzate con i valori reali al momento dell'uso.
- Tutte le variabili legate ai cicli devono essere reinizializzate con l'entrata in una nuova iterazione prima di essere riutilizzate nel nuovo ciclo.
- Tutte le strutture devono essere azzerate prima del loro utilizzo.
- Tutti i buffer devono essere azzerati prima del loro utilizzo o riutilizzo.

7.1.8.2 Utilizzo dei tipi di dati

Stringhe

- Tutte le stringhe devono essere terminate dal carattere NULL. Evitare errori logici di programmazione che agevolino l'insorgere di una condizioni di memory leak. Deve essere riposta la massima attenzione nell'utilizzo di funzioni che non aggiungono al termine di una stringa copiata in un buffer di destinazione il carattere NULL se questo non risiede nel buffer sorgente.
  Esempio:
  Forma non corretta:
  ```
  strncpy(dest, source, sizeof(dest));
  ```
  Forma corretta:
  ```
  strncpy(dest, source, sizeof(dest);
  dest[sizeof(dest) – 1] = '\0';
  ```

- Il codice non deve effettuare operazioni su una stringa o su un char arrayche non siano terminati dal carattere NULL;
- L'input proveniente dall'utente deve sempre essere convalidato e scremato da caratteri invalidi (; | ! & ~ ' " - * % ` \ / < > ? $ @ : ( ) [ ] { } .) prima di essere passato alle successive elaborazioni dell'applicazione (ad esempio alla funzione system());
- Utilizzare le funzioni strspn(), strcspn() e strpbrk() per filtrare l'input utente;
- Il formato delle stringhe deve sempre essere specificato nei parametri delle funzioni che lo richiedono. In questo contesto le funzioni considerate critiche e soggette a problematiche di format string overflow, se non correttamente utilizzate, sono: printf(), fprintf(), sprintf(), snprintf(), vprintf(), vfprintf(), vsprintf(), vsnprintf(), scanf(), fscanf(), sscanf(), vscanf(), vsscanf(), vfscanf(), wprintf(), fwprintf(), swprintf(), vwprintf(), vfwprintf(), vswprintf().
  Esempio:
  Forma non corretta:
  ```
  printf(buffer1);
  snprintf(dest, sizeof(dest), buf);
  fprintf(FILE, num, stringa);
  ```
  Forma corretta:
  ```
  printf("%s\r\n", buffer1);
  snprintf(dest, sizeof(dest), "%s", buf);
  fprintf(FILE, "%d: %s%\n", num, stringa);
  ```

Buffer

Tutti i buffer devono essere abbastanza grandi per contenere i dati a loro destinati, inoltre:

- Evitare l'utilizzo di funzioni che non consentono di specificare la dimensione delle stringhe copiate da un buffer sorgente a uno di destinazione. Le funzioni considerate critiche in questo contesto, che non devono mai essere utilizzate sono: strcpy(), wcscpy(), sprintf(), strcat(), gets(), scanf(), vsprintf() e wcscat();
- Quando i dati vengono copiati all'interno di un buffer deve essere sempre verificata la loro dimensione confrontandola con quella del buffer di destinazione. Le funzioni considerate critiche per errori di bound-checking, pur permettendo di specificare la lunghezza delle stringhe soggette a copia da un buffer all'altro, sono: strncpy(), wcsncpy(), snprintf(), strncat(), vsnprintf(), wcsncat(),