Memory Leak

A memory leak, in computer programming, occurs when a computer program consumes memory but is unable to release it back to the operating system.

A memory leak can diminish the performance of the computer by reducing the amount of available memory. Eventually, in the worst case, too much of the available memory may become allocated and all or part of the system or device stops working correctly, the application fails, or the system slows down unacceptably due to thrashing.

Memory leaks may not be serious or even detectable by normal means. In modern operating systems, normal memory used by an application is released when the application terminates. This means that a memory leak in a program that only runs for a short time may not be noticed and is rarely serious.

Example of Memory Leak in C

```
#include <stdlib.h>
void function_which_allocates(void)
{
   /* allocate an array of 45 floats */
   float *a = malloc(sizeof(float) * 45);
   /* additional code making use of 'a' */
   /* return to main, having forgotten to free the memory we malloc'd */
}
int main(void)
{
   function_which_allocates();
   /* the pointer 'a' no longer exists, and therefore cannot be freed,
    but the memory is still allocated. a leak has occurred. */
}
```

We have to use the free() function to release the memory referenced by the pointer *a to avoid any memory leak.

Macros v/s Functions

Macros are just substitution patterns applied to your code. They can be used almost anywhere in your code because they are replaced with their expansions before any compilation starts.

Functions, on the other hand, are actual block of code whose body is directly injected into the call site.

Some points to note:

- In C, macro invocations do not perform type checking, or even check that arguments are well-formed, whereas function calls usually do.
- In C, a macro cannot use the return keyword with the same meaning as a function would do (it would make the function that asked the expansion terminate, rather than the macro). In other words, a macro cannot return anything which is not the result of the last expression invoked inside it.
- Since C macros use mere textual substitution, this may result in unintended side-effects and inefficiency due to re-evaluation of arguments and order of operations.
- Compiler errors within macros are often difficult to understand, because they refer to the expanded code, rather than the code the programmer typed. Thus, debugging information for inline code is usually more helpful than that of macro-expanded code.
- Many constructs are awkward or impossible to express using macros, or use a significantly different syntax.
- Many compilers can also inline expand some recursive functions; recursive macros are typically illegal.

Pseudocode

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are not essential for human understanding of the algorithm, such as variable declarations, system-specific code and some functions.

The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program. Pseudocode resembles, but should not be confused with skeleton programs, including dummy code, which can be compiled without errors. Flowcharts can be thought of as a graphical alternative to pseudocode, but are more spacious on paper.

C Style Pseudocode

```
void function fizzbuzz

For (i = 1; i<=100; i++) {
    set print_number to true;
    If i is divisible by 3
        print "Fizz";
        set print_number to false;
    If i is divisible by 5
        print "Buzz";
        set print_number to false;
    If print_number, print i;
    print a newline;
}
```

Difference between Algorithm and Pseudo-code

Algorithm --> A precise rule (or set of rules) specifying how to solve some problem

Algorithm means a method to solve the given problem.

Pseudo-Code --> Statements outlining the operation of a computer program, written in something similar to computer language but in a more understandable format

Pseudo-code is a way to describe the algorithm in order to transform the algorithm into real source code. You can write a Pseudo-code in any way if you like. But mostly, we use widely accepted symbols to write it.

In short Pseudocode does not have any Rules its some kind of note making that helps us to tackle the program. Algorithm have some rules and logic.