How Stack Memory Works

A stack (call stack) works in the order of last in first out or first in last out. A stack stores frames, symbols, addresses, and values. A stack frame is formed when a function is called. After a function is called the return location is pushed in the frame of the called function, for instance if program A calls program B, the return location is pushed in the frame of program B. The return location is always the first thing pushed in the frame of a called function. If a function is called more than once, each call has its corresponding return location. If a variable is to store the return value of a called function, the variable would be given “garbage” value in the beginning. After the called function returns its value, the value of the variable will be modified from “garbage” to the return value. The address of such variable (one to store the return value of a called function) is called the value address. The value address is pushed in the stack frame of the called function directly above the return location. After the value address is pushed, if a called function has arguments, they are pushed in the stack frame of the called function left to right, meaning the left most variable is pushed is pushed first and then the next and so on until all the arguments are pushed. After the arguments are pushed, the local variables of the called function are pushed top to bottom, meaning the top local variable is pushed first and then the next and so on until they are all pushed. Also, an array is pushed on the stack element by element. The first pushed array element is the one at index 0, then the element at index 1, after that the one at index 2, and so on until the array elements are all pushed.