Call Stack

→ A region of memory allocated for every function call

→ A stack frame is the allocated region of a call stack every time a function is called.

→ Variables/Parameters defined within a function are stored in the stack frame created by calling a function.

When a function returns a value/exits, the stack frame will be destroyed (memory freed up to be reallocated)

Stack frame goes from bottom to top (stack frame of most recently called function is at the top of the call stack).

Arrays

Compound data type - Stores multiple values of same type

Declaring in array

type of element
we want to
store   name_of_array [number of elements you want to store]

Unspecified elements in an array are initialized to zero.

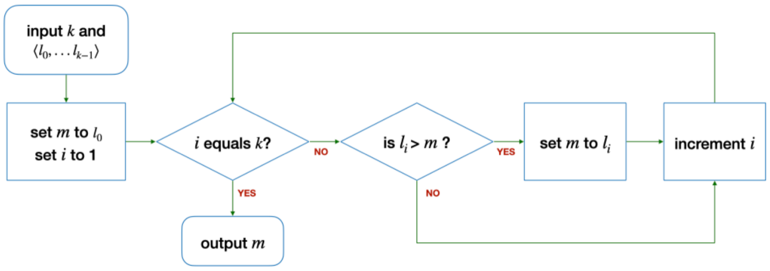
To initialize an array, declare and initialize on the same line.

LONG_MAX => infinity

Arrays as a function parameter

```
output type fn_name(array elem type array_name[array size]){...}
output type fn_name(long arr_len, array elem type array_name[arr_len])
```

Find max value within an array



input k and $\langle l_0, \dots l_{k-1} \rangle$
set m to l_0 set i to 1
i equals k?
NO
YES
output m
is $l_i > m$?
YES
NO
set m to l_i
increment i

```
long max(long k, long list[k]){
    long m = list[0];
    for (long i=1; i<k; i+=1){
        if (m < list[i]){
            m = list[i];
        }
    }
    return m;
}
```

Processing of arrays

When calling an array in a function in C, rather than creating a copy of the array it refers to the memory address of the array and modifies data there.

What is copied onto the functions' stack frame is the memory address of the array rather than the array itself. - "PASS BY REFERENCE" const keyword indicateds variable is read-only (i.e cannot be modified)

Pointers

Variable that contains the memory address of another variable

Declare a pointer variable using the asterik (*) operator. (Eg double *ptr;)

Array decay

Referring to just the name of an array is equivalent to referring to the memory address of its first element.

```
long list[10];
```

list "decays" into the memory address of its first element.

In order to return the memory address of the whole array the function should be typed as output type* fn-name.

Types of arrays

Fixed length arrays

Variable length arrays

Dynamically Allocated arrays