```
/* pointers and memory management */

/**
 * pointers
 */
err error; ptr i8* = nullptr

ptr, err = alloc(1024) // allocate 1KB on the heap
defer ptr          // binds the pointer to the scope
                   // it deletes the memory when scope ends

value i8 = *ptr   // the value it points to
ptr1 := &value    // ptr1 is the same as ptr
delete(ptr)        // you can also manually delete the memory

/**
 * pointer arithmetic
 */

// pointers can only be added or substracted between them.
// pointes arithmetics results are always in 64 bits values.

// example: lets say amount_of_i8 is 6, this means 6 bytes
amount_of_i8 i64 = ptr1 - ptr

// pointers can be incremented or decremented. This is done in sizeof(type) where
// type is the typed pointed by the pointer

ptr_to_i64 i64* = &variable // here ptr_to_i64 is (e.g): 0x7CF7
ptr_to_i64++  // here is incremented by 8bytes (64bits): 0x7CFF
ptr_to_i64 + 1  // here is also incremented by 8bytes (64bits): 0x7D07

// if you want to move in bytes you should:
ptr_to_i8 i8* = cast(i8*, &variable)
ptr_to_i8++ // now is incremented by 1byte (8bits)

/**
 * pointers as params
 */
fn my_fn(ptr_param u32*) {
  /* this is great for passing a bigger than 64 bits types
   * NOTE: remember you will not be able to modify
   * the value of the pointer nor the data it points to */
}
```