Parker Dunn

April 23rd, 2021

# EK125 - Exam 3 Review

## Chapter 3 - "Cheat Sheet"

**TOPICS:** → ~~Arrays~~ Data Structures: Arrays & strings

→ Data structures: Structs, Nested Structs, struct arrays

**Arrays:**

(*) Declaration → # of elements in brackets! → Declaring → start @ 1

@ Use → # of indices used in brackets! → Indexes → start @ 0

(*) size of arrays cannot change after being declared!

Example of Two Dimensional Array:

Declaration → `int numbers [4][4];` → 4 x 4 array of ints

Indexing → `numbers [row][column];` → where $0 \leq row \leq 3$, $0 \leq column \leq 3$

**Strings**

→ "X" ≡ 'X \0'

↳ TWO CHARACTERS = (*) Declaration char (must be @ least 2) X[2] = 'X \0'

→ Declarations are legitimately just arrays of characters!

→ String assignments (e.g. str = "Parker") only @ declaration!

→ String assignment options (after declaration)

(*) ① strcopy - e.g. strcopy (strvariable, "Parker");

② scanf - cumbersome

③ ~~gets~~/fgets - e.g. fgets (newstring (variable), ); ↳ gets an input

(*) → strcat - e.g. strcat (strvar, str) - concatenates str to strvar

(*) → Strlen → returns the actual # of characters stored in the string NOT INCLUDING '\0' (string delimiter)

# Structures

## Array of structures (EXAMPLE)

Valid Expressions

| EXPRESSION | What it is | Type |
|---|---|---|
| packages | the entire array | item_t [] |
| packages[1] | structure with all members | (struct) item_t |
| packages[0].cost | cost member of structure @ index 0 | float |
| packages[2].code | code member of structure @ index 2 | char |

① Define:

```
typedef struct
{
   int item_no;
   float cost;
   float price;
   char code;
} item_t;
```

② Declare:

item_t packages[4]

↳ array of structures = packages

↳ # of elements: 4

③ Define:

Skipped writing out a full example

↳ ① Must be done in a chunk (multiple rows and cols) @ declaration

↳ ② Assign individual array elements

Parker Dunn

April 23rd 2021

# EK 125 – Exam 3 Review

## Chapter 4 – "Cheat Sheet"

Topics → User defined functions
→ Addresses
→ Pointers

### User Defined Functions :

① Return values w/ RETURN STATEMENT

→ "~~ed~~ addresses of variables are passed to a function so it can change the value of the variable" → CALL-BY-REFERENCE

→ Diagram of call-by-value (to demonstrate the need for call by reference)

Function: take_these → recieves two inputs ① float and ② int

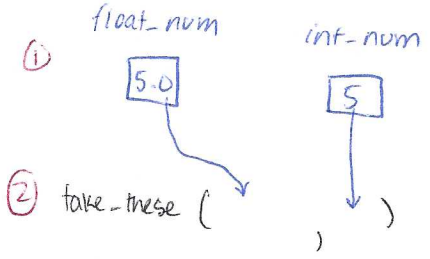
**In main()**

* function call *

~~* take_tese~~

take_these ( float_num, int_num)

What that looks like...

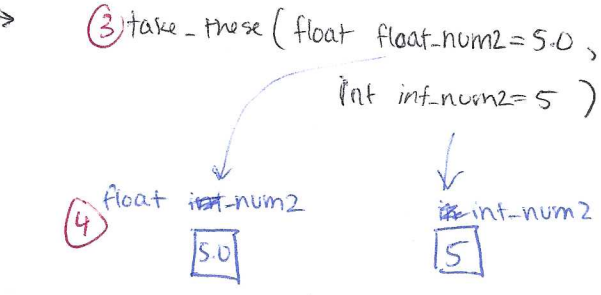



**The function**

challenging ~~challenging~~ intentionally used same name to show a confusing example

~~void~~

```
void take_these ( float float_num2, int ~~int_num~~ int_num2 )
{
  // content here
}
```




The order of assignment/ events in RED.

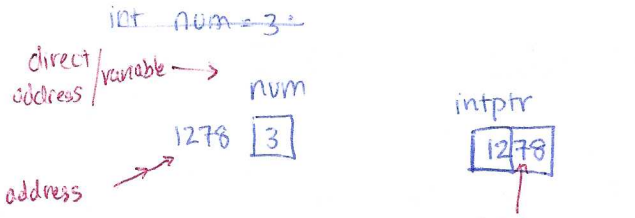
NOTICE: The values are passed, but the original variables are NOT @ the SAME MEMORY LOCATION

## Address & Pointer Diagram : Two equivalent representations

CONTEXT: ① num = 3 & num is an int

② intptr points to an int ( * int *iptr = declaration)

① Pointer Variables using "Address"




→ *intptr

↳ goes to address 1278

↳ allows use of 3

How to think about it

② Pointer variables using Pointers


→ *intptr

↳ is the actual value at the end of the pointer

How to think about it

"Book" Overview

## Chapter 2: Selection & Loop Statements

### CHAPTER 2A - Selection Statements

(I.) 2.1 Compound and Empty statements

~~(II.)~~ { } vs. ;

(II.) 2.2 Selection statements

2.2.1 IF and IF-ELSE Statements

- → Boolean Expressions (a little)
- → Rational operators (comparisons between same type)
- → Logical operators (i.e. || → "or")

(*) → p.4 ORDER OF PRECEDENCE

2.2.2 The switch Statement

- → Switch & (cases) statements
- → switch expression → must be an integral expression (int or char type)
- → BREAK!
- → "Default" (not otherwise)

### CHAPTER 2B - LOOP STATEMENTS

(I.) 2.3 Loop Statements

2.3.1 The For loop

- → Step value
- → Loops may be nested ⇒ Nested For loops

Overview

# Chapter 2: Selection and Loops (cont.)

## 2.3.2. The While and Do-while loops

→ "CONDITIONAL" loops

→ scanf error checking

→ scanf as a conditional

↳ # of entries received is returned!

## 2.3.3 Flushing the Input Buffer and Error-checking

→ fflush

→ ~~Attempt~~ Coding flush of input buffer:

Built in C function... I think

Universal form of flushing input buffer

```
while ((let = getchar()) != '\n' &&
                        let != EOF);
```

→ ⊛ From experience:

correct Syntax for fflush: `fflush(stdin);` → "flush the standard input"

*KNOWN*

doesn't need to be declared

→ Error checking

## (II) 2.4 Application of Loops: File Input/Output

→ FOUR STEPS FOR WORKING W/ FILES
① Declare FILE variable
② Open file
③ Read, write, append
④ Close

→ fscanf function

→ Type: FILE (all caps)

→ Brief intro to * ("pointer") variables

→ fopen function

→ fclose function

→ ~~POINTER~~ FILE POINTERS INITIALIZED IN STDIO.h

Ex. stdin (standard input) & stdio (standard output)

⊛ fprintf and fscanf are in stdio.h

# Chapter 3: Data Structures

## Chapter 3A: Data Structures – Arrays & Strings

(*) C has
1. strings
2. character arrays
3. arrays (vectors & matrices)
4. structures

(*) C DOES NOT have:
1. cell arrays
2. ... or the other elaborate data structures from MATLAB

## (I.) 3.1 Arrays

→ No built in functions! – Lots of looping

### 3.1.1 One Dimensional Arrays

→ Indices vs. declaring
↓
start @ 0

→ declare arrays w/ { } (actual values) | declare size w/ [ ]

→ Index with [ ]

→ Arrays ≡ "for loop" → no built in functions!
↳ No shortcut for creation!

→ Size of arrays cannot change
→ For Loops for populating arrays

(*) Declaring more array elements than you need!
↳ While loops w/ arrays!

(*) ↳ using "last" variable when the entirety of an array is used!

↳ Error checking a variety of things
1. Not entering ~~too many~~ values beyond array bounds
2. ~~The~~ An ~~input~~ input ~~for~~ before storing in an array

BOOK Review

(Chapter 3A cont.)

3.1.2 Two Dimensional Arrays

Declaration → int numbers [4][4]; → 4×4 array

Indexing → number [row][column] → where ~~row~~

$0 \leq row \leq 3$

$0 \leq column \leq 3$

TWO DIMENSIONAL ARRAY ≡ "Nested for loops"

↳ Order of nested loops

(II) 3.2 STRINGS

→ "ARRAYS OF CHARACTERS"

↳ null character → '\0'

↳ a.k.a "string delimiter" AND "end-of-string" sentinal

→ Declaring string variables = char arrays

↳ e.g. new str [20] → 19 chars + 1 '\0'

→ String declarations, String assignments (* tricky *)

Initializing strings

① Reading into a string

→ scanf → ⊗ NO ~~ADDRESS~~ AMPERSAND & to scanf for char array

→ gets - "get string"

(& fgets)

② String functions

→ String.h → HEADER

→ strcopy

→ strcat

→ strlen

→ strcmp → 0 = identical

e.g. strcmp(str1,str2) 1 = str1 > str2

-1 = str1 < str2

Book Outline

# Chapter 3B: Data Structures – Structures

(I.) 3.3 ~~Arrays~~ Structures

→ overview of structured variables / data structures
  ↳ logically related data
  ↳ different types of data

→ "Struct" syntax → does not show how to reference an "unnamed" structure! ← w/o an alias!

→ Intro to TYPEDEF

~~→~~ ↳ Typedef for structures

## 3.3.1 Arrays of structures

→ Accessing structures of the array

→ "dot" referencing structure members

→ Looping through a structure array

~~→~~

## 3.3.2 Nested Structures

→ "~~double dot two~~ Double "dot operator" notation

→ using two structure types

~~→~~