0x07. C - Even more pointers, arrays and strings

**C**

* By: Julien Barbier
* Weight: 1
* Project over - took place from Sep 26, 2022 6:00 AM to Sep 27, 2022 6:00 AM
* An auto review will be launched at the deadline

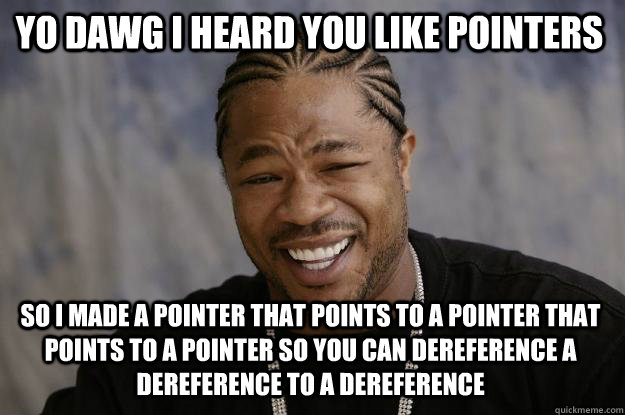
In a nutshell…

* **Auto QA review:** 42.0/56 mandatory & 0.0/17 optional
* **Altogether:**  **75.0%**
  + Mandatory: 75.0%
  + Optional: 0.0%
  + Calculation:  75.0% + (75.0% \* 0.0%)  == **75.0%**

Concepts

*For this project, we expect you to look at this concept:*

* [Pointers and arrays](https://intranet.alxswe.com/concepts/60)



Resources

**Read or watch**:

* [C - Pointer to Pointer](https://intranet.alxswe.com/rltoken/eyikXPg7ZxCAEuWklB6xtQ)
* [C – Pointer to Pointer with example](https://intranet.alxswe.com/rltoken/ojr7OUUm2I-MULE4lWlrkg)
* [Multi-dimensional Arrays in C](https://intranet.alxswe.com/rltoken/HUZIJ6t55KM7d7FBCwWm8Q)
* [Two dimensional (2D) arrays in C programming with example](https://intranet.alxswe.com/rltoken/Dx9nIBRj68sRBGe2NRI_aQ)

Learning Objectives

At the end of this project, you are expected to be able to [explain to anyone](https://intranet.alxswe.com/rltoken/YpzhlccIJNihbnYgObEStg), **without the help of Google**:

General

* What are pointers to pointers and how to use them
* What are multidimensional arrays and how to use them
* What are the most common C standard library functions to manipulate strings

Copyright - Plagiarism

* You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
* You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
* You are not allowed to publish any content of this project.
* Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

* Allowed editors: vi, vim, emacs
* All your files will be compiled on Ubuntu 20.04 LTS using gcc, using the options -Wall -Werror -Wextra -pedantic -std=gnu89
* All your files should end with a new line
* A README.md file, at the root of the folder of the project is mandatory
* Your code should use the Betty style. It will be checked using [betty-style.pl](https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and [betty-doc.pl](https://github.com/holbertonschool/Betty/blob/master/betty-doc.pl)
* You are not allowed to use global variables
* No more than 5 functions per file
* You are not allowed to use the standard library. Any use of functions like printf, puts, etc… is forbidden
* You are allowed to use [\_putchar](https://github.com/holbertonschool/_putchar.c/blob/master/_putchar.c)
* You don’t have to push \_putchar.c, we will use our file. If you do it won’t be taken into account
* In the following examples, the main.c files are shown as examples. You can use them to test your functions, but you don’t have to push them to your repo (if you do we won’t take them into account). We will use our own main.c files at compilation. Our main.c files might be different from the one shown in the examples
* The prototypes of all your functions and the prototype of the function \_putchar should be included in your header file called main.h
* Don’t forget to push your header file

More Info

You do not need to learn about pointers to functions, arrays of structures, malloc and free - yet.

Quiz questions

**Great!** You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. memset

**mandatory**

Score: 85.71% (*Checks completed: 85.71%*)

Write a function that fills memory with a constant byte.

* Prototype: char \*\_memset(char \*s, char b, unsigned int n);
* The \_memset() function fills the first n bytes of the memory area pointed to by s with the constant byte b
* Returns a pointer to the memory area s

FYI: The standard library provides a similar function: memset. Run man memset to learn more.

julien@ubuntu:~/0x07$ cat 0-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* simple\_print\_buffer - prints buffer in hexa

\* @buffer: the address of memory to print

\* @size: the size of the memory to print

\*

\* Return: Nothing.

\*/

void simple\_print\_buffer(char \*buffer, unsigned int size)

{

unsigned int i;

i = 0;

while (i < size)

{

if (i % 10)

{

printf(" ");

}

if (!(i % 10) && i)

{

printf("\n");

}

printf("0x%02x", buffer[i]);

i++;

}

printf("\n");

}

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char buffer[98] = {0x00};

simple\_print\_buffer(buffer, 98);

\_memset(buffer, 0x01, 95);

printf("-------------------------------------------------\n");

simple\_print\_buffer(buffer, 98);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main.c 0-memset.c -o 0-memset

julien@ubuntu:~/0x07$ ./0-memset

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

-------------------------------------------------

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x00 0x00 0x00

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 0-memset.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

1. memcpy

**mandatory**

Score: 85.71% (*Checks completed: 85.71%*)

Write a function that copies memory area.

* Prototype: char \*\_memcpy(char \*dest, char \*src, unsigned int n);
* The \_memcpy() function copies n bytes from memory area src to memory area dest
* Returns a pointer to dest

FYI: The standard library provides a similar function: memcpy. Run man memcpy to learn more.

julien@ubuntu:~/0x07$ cat 1-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* simple\_print\_buffer - prints buffer in hexa

\* @buffer: the address of memory to print

\* @size: the size of the memory to print

\*

\* Return: Nothing.

\*/

void simple\_print\_buffer(char \*buffer, unsigned int size)

{

unsigned int i;

i = 0;

while (i < size)

{

if (i % 10)

{

printf(" ");

}

if (!(i % 10) && i)

{

printf("\n");

}

printf("0x%02x", buffer[i]);

i++;

}

printf("\n");

}

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char buffer[98] = {0};

char buffer2[98] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14};

simple\_print\_buffer(buffer, 98);

\_memcpy(buffer + 50, buffer2, 10);

printf("-------------------------------------------------\n");

simple\_print\_buffer(buffer, 98);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main.c 1-memcpy.c -o 1-memcpy

julien@ubuntu:~/0x07$ ./1-memcpy

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

-------------------------------------------------

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x01 0x02 0x03 0x04 0x05 0x07 0x07 0x08 0x09 0x0a

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 1-memcpy.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

2. strchr

**mandatory**

Score: 85.71% (*Checks completed: 85.71%*)

Write a function that locates a character in a string.

* Prototype: char \*\_strchr(char \*s, char c);
* Returns a pointer to the first occurrence of the character c in the string s, or NULL if the character is not found

FYI: The standard library provides a similar function: strchr. Run man strchr to learn more.

julien@ubuntu:~/0x07$ cat 2-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char \*s = "hello";

char \*f;

f = \_strchr(s, 'l');

if (f != NULL)

{

printf("%s\n", f);

}

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main.c 2-strchr.c -o 2-strchr

julien@ubuntu:~/0x07$ ./2-strchr

llo

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 2-strchr.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

3. strspn

**mandatory**

Score: 85.71% (*Checks completed: 85.71%*)

Write a function that gets the length of a prefix substring.

* Prototype: unsigned int \_strspn(char \*s, char \*accept);
* Returns the number of bytes in the initial segment of s which consist only of bytes from accept

FYI: The standard library provides a similar function: strspn. Run man strspn to learn more.

julien@ubuntu:~/0x07$ cat 3-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char \*s = "hello, world";

char \*f = "oleh";

unsigned int n;

n = \_strspn(s, f);

printf("%u\n", n);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-main.c 3-strspn.c -o 3-strspn

julien@ubuntu:~/0x07$ ./3-strspn

5

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 3-strspn.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

4. strpbrk

**mandatory**

Score: 85.71% (*Checks completed: 85.71%*)

Write a function that searches a string for any of a set of bytes.

* Prototype: char \*\_strpbrk(char \*s, char \*accept);
* The \_strpbrk() function locates the first occurrence in the string s of any of the bytes in the string accept
* Returns a pointer to the byte in s that matches one of the bytes in accept, or NULL if no such byte is found

FYI: The standard library provides a similar function: strpbrk. Run man strpbrk to learn more.

julien@ubuntu:~/0x07$ cat 4-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char \*s = "hello, world";

char \*f = "world";

char \*t;

t = \_strpbrk(s, f);

printf("%s\n", t);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-main.c 4-strpbrk.c -o 4-strpbrk

julien@ubuntu:~/0x07$ ./4-strpbrk

llo, world

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 4-strpbrk.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

5. strstr

**mandatory**

Score: 85.71% (*Checks completed: 85.71%*)

Write a function that locates a substring.

* Prototype: char \*\_strstr(char \*haystack, char \*needle);
* The \_strstr() function finds the first occurrence of the substring needle in the string haystack. The terminating null bytes (\0) are not compared
* Returns a pointer to the beginning of the located substring, or NULL if the substring is not found.

FYI: The standard library provides a similar function: strstr. Run man strstr to learn more.

julien@ubuntu:~/0x07$ cat 5-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char \*s = "hello, world";

char \*f = "world";

char \*t;

t = \_strstr(s, f);

printf("%s\n", t);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-main.c 5-strstr.c -o 5-strstr

julien@ubuntu:~/0x07$ ./5-strstr

world

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 5-strstr.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

6. Chess is mental torture

**mandatory**

Score: 0.0% (*Checks completed: 0.0%*)

Write a function that prints the chessboard.

* Prototype: void print\_chessboard(char (\*a)[8]);

julien@ubuntu:~/0x07$ cat 7-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char board[8][8] = {

{'r', 'k', 'b', 'q', 'k', 'b', 'k', 'r'},

{'p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'},

{' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},

{'P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'},

{'R', 'K', 'B', 'Q', 'K', 'B', 'K', 'R'},

};

print\_chessboard(board);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 \_putchar.c 7-main.c 7-print\_chessboard.c -o 7-print\_chessboard

julien@ubuntu:~/0x07$ ./7-print\_chessboard

rkbqkbkr

pppppppp

PPPPPPPP

RKBQKBKR

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 7-print\_chessboard.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

7. The line of life is a ragged diagonal between duty and desire

**mandatory**

Score: 85.71% (*Checks completed: 85.71%*)

Write a function that prints the sum of the two diagonals of a square matrix of integers.

* Prototype: void print\_diagsums(int \*a, int size);
* Format: see example
* You are allowed to use the standard library

Note that in the following example we are casting an int[][] into an int\*. This is not something you should do. The goal here is to make sure you understand how an array of array is stored in memory.

julien@ubuntu:~/0x07$ cat 8-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

int c3[3][3] = {

{0, 1, 5},

{10, 11, 12},

{1000, 101, 102},

};

int c5[5][5] = {

{0, 1, 5, 12124, 1234},

{10, 11, 12, 123521, 12512},

{1000, 101, 102, 12545, 214543435},

{100, 1012451, 11102, 12545, 214543435},

{10, 12401, 10452, 11542545, 1214543435},

};

print\_diagsums((int \*)c3, 3);

print\_diagsums((int \*)c5, 5);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 8-main.c 8-print\_diagsums.c -o 8-print\_diagsums

julien@ubuntu:~/0x07$ ./8-print\_diagsums

113, 1016

1214556093, 1137318

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 8-print\_diagsums.c

 Done? Help Check your code Ask for a new correction Get a sandbox QA Review

8. Double pointer, double fun

**#advanced**

Score: 0.0% (*Checks completed: 0.0%*)

Write a function that sets the value of a pointer to a char.

* Prototype: void set\_string(char \*\*s, char \*to);

julien@ubuntu:~/0x07$ cat 100-main.c

#include "main.h"

#include <stdio.h>

/\*\*

\* main - check the code

\*

\* Return: Always 0.

\*/

int main(void)

{

char \*s0 = "Bob Dylan";

char \*s1 = "Robert Allen";

printf("%s, %s\n", s0, s1);

set\_string(&s1, s0);

printf("%s, %s\n", s0, s1);

return (0);

}

julien@ubuntu:~/0x07$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-main.c 100-set\_string.c -o 100-set\_string

julien@ubuntu:~/0x07$ ./100-set\_string

Bob Dylan, Robert Allen

Bob Dylan, Bob Dylan

julien@ubuntu:~/0x07$

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 100-set\_string.c

 Done? Help Check your code Ask for a new correction QA Review

9. My primary goal of hacking was the intellectual curiosity, the seduction of adventure

**#advanced**

Score: 0.0% (*Checks completed: 0.0%*)

Create a file that contains the password for the [crackme2](https://github.com/holbertonschool/0x06.c) executable.

* Your file should contain the exact password, no new line, no extra space
* ltrace, ldd, gdb and objdump can help
* You may need to install the openssl library to run the crakme2 program: sudo apt install libssl-dev
* Edit the source list sudo nano /etc/apt/sources.list to add the following line: deb http://security.ubuntu.com/ubuntu xenial-security main Then sudo apt update and sudo apt install libssl1.0.0

**Repo:**

* GitHub repository: alx-low\_level\_programming
* Directory: 0x07-pointers\_arrays\_strings
* File: 101-crackme\_password