(/)

0x05. C - Pointers, arrays and strings

C

By Julien Barbier

Weight: 1

Project over - took place from Jul 13, 2022 to Jul 14, 2022 - you're done with 200% of tasks.

An auto review will be launched at the deadline

In a nutshell...

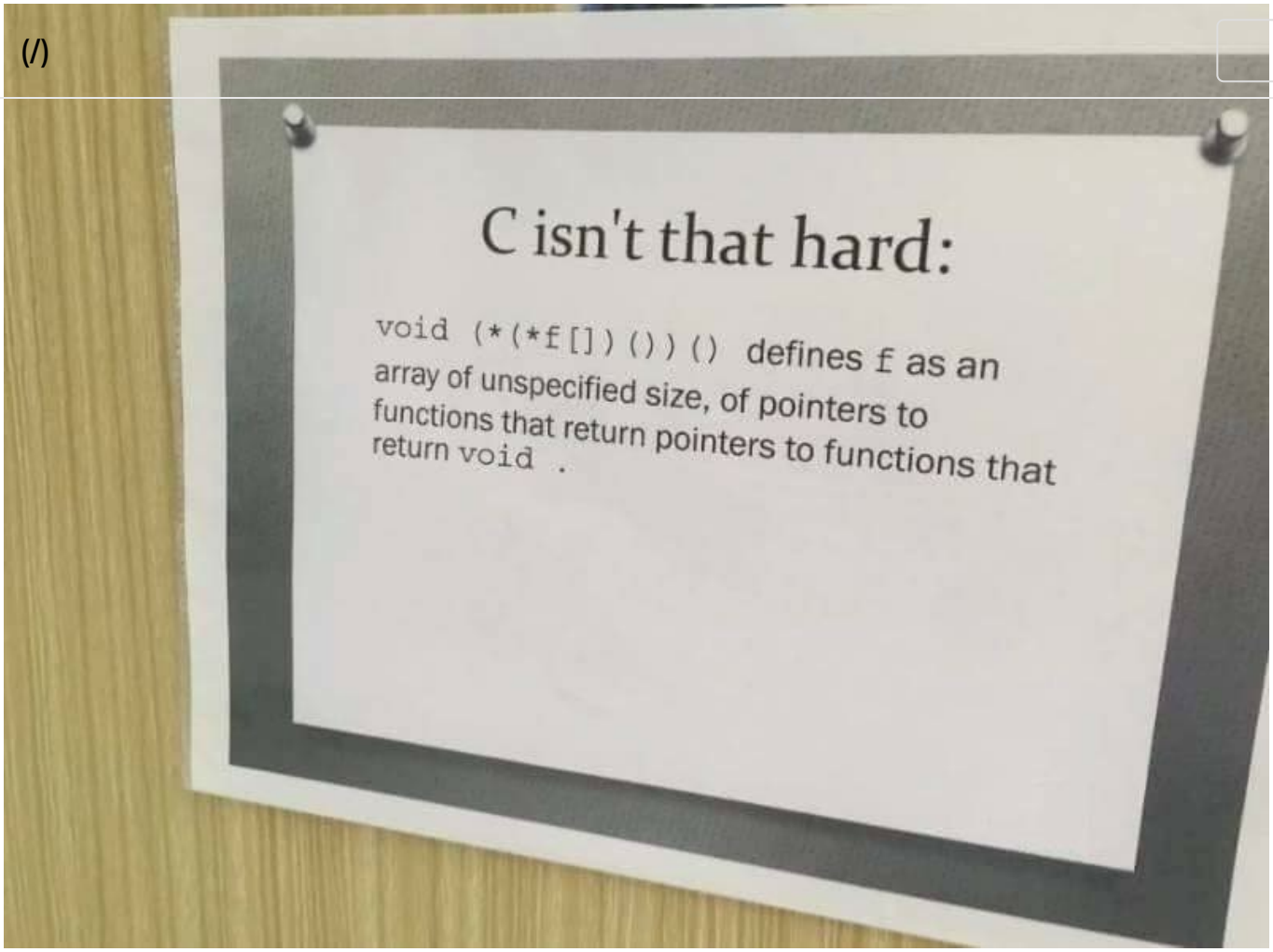
- **Auto QA review:** 71.0/71 mandatory & 24.0/24 optional
- **Altogether: 200.0%**
  - Mandatory: 100.0%
  - Optional: 100.0%
  - Calculation: 100.0% + (100.0% * 100.0%) == **200.0%**

Concepts

For this project, we expect you to look at these concepts:

- Pointers and arrays (/concepts/60)
- Data Structures (/concepts/120)

(/)



Resources

Read or watch:

- C - Arrays (/rltoken/PVi2XMuApOK3jfhsoqsyXw)
- C - Pointers (/rltoken/oyHybzYBeFiLUMALpb_usA)
- C - Strings (/rltoken/sUeh9qDyW9pePOfJlpx_Bw)
- Memory Layout (/rltoken/0k6CD2ZMzSFOMUxMOBiAIQ)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/OLGzlaD19ia5NZ-WCMckeg), **without the help of Google**:

General

- What are pointers and how to use them
- What are arrays and how to use them
- What are the differences between pointers and arrays
- How to use strings and how to manipulate them
- Scope of variables

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, pointers to pointers, multidimensional arrays, arrays of structures, `malloc` and `free` - yet.

Quiz questions

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

(/)

Question #0

What is the size of a pointer to a `char` (on a 64-bit architecture)

- [ ] 1 byte
- [ ] 2 bytes
- [ ] 4 bytes
- [x] 8 bytes

Question #1

What is the size of a pointer to an `int` (on a 64-bit architecture)

- [ ] 1 byte
- [ ] 2 bytes
- [ ] 4 bytes
- [x] 8 bytes

Question #2

If we have a variable called `var` of type `int`, how can we get its address in memory?

- [ ] *var
- [ ] *(var)
- [x] &var

Question #3

What is the identifier to print an address with `printf`?

- [ ] %a
- [ ] %d
- [x] %p
- [ ] %x

Question #4

The process of getting the value that is stored in the memory location pointed to by a pointer is called:

(/)

- [ ] Pointing
- [ ] Accessing
- [x] Dereferencing
- [ ] Casting

Question #5

Is it possible to declare a pointer to a pointer?

- [x] Yes
- [ ] No
- [ ] It depends on the type the pointer is pointing to

Question #6

What happens when one tries to access an illegal memory location?

- [ ] The operation is ignored
- [x] Segmentation fault
- [ ] The computer shuts down
- [ ] There's a chance for the computer to catch fire, and sometimes even explode

Question #7

What is the value of `n` after the following code is executed?

```
int n = 98;
int *p = &n;
```

- [ ] 0
- [x] 98
- [ ] 99
- [ ] 402

Question #8

What is the value of `n` after the following code is executed?

```
int n = 98;
int *p = &n;

p = 402;
```

- [ ] 0
- [x] 98
- [ ] 99
- [ ] 402

Question #9

What is the value of `n` after the following code is executed?

```
int n = 98;
int *p = &n;

*p = 402;
```

- [ ] 0
- [ ] 98
- [ ] 99
- [x] 402

Question #10

What is the value of `n` after the following code is executed?

```
int n = 98;
int *p = &n;

*p++;
```

- [ ] 0
- [x] 98
- [ ] 99
- [ ] 402

Question #11

We declare the following variable

```
int arr[5];
```

What is the size in memory of the variable `arr` ?

- [ ] 4 bytes
- [ ] 5 bytes
- [ ] 8 bytes
- [ ] 10 bytes
- [ ] 32 bytes
- [x] 20 bytes

Question #12

We declare the following variable

```
int arr[5];
```

What is the equivalent of typing `arr[2]` ?

- [ ] arr + 2
- [ ] *arr + 2
- [x] *(arr + 2)

Tasks

0. 98 Battery st.

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that takes a pointer to an `int` as parameter and updates the value it points to to `98` .

- Prototype: `void reset_to_98(int *n);`

```
julien@ubuntu:~/0x05$ cat 0-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int n;

    n = 402;
    printf("n=%d\n", n);
    reset_to_98(&n);
    printf("n=%d\n", n);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-main.c 0-rese
t_to_98.c -o 0-98
julien@ubuntu:~/0x05$ ./0-98
n=402
n=98
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `0-reset_to_98.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

1. Don't swap horses in crossing a stream

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that swaps the values of two integers.

- Prototype: `void swap_int(int *a, int *b);`

```
julien@ubuntu:~/0x05$ cat 1-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int a;
    int b;

    a = 98;
    b = 42;
    printf("a=%d, b=%d\n", a, b);
    swap_int(&a, &b);
    printf("a=%d, b=%d\n", a, b);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-main.c 1-swa
p.c -o 1-swap
julien@ubuntu:~/0x05$ ./1-swap
a=98, b=42
a=42, b=98
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `1-swap.c`

Done! | Help | Check your code | QA Review

2. This report, by its very length, defends itself against the risk of being read

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that returns the length of a string.

- Prototype: `int _strlen(char *s);`

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

```
julien@ubuntu:~/0x05$ cat 2-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;
    int len;

    str = "My first strlen!";
    len = _strlen(str);
    printf("%d\n", len);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-main.c 2-strl
en.c -o 2-strlen
julien@ubuntu:~/0x05$ ./2-strlen
16
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `2-strlen.c`

Done! | Help | Check your code | QA Review

3. I do not fear computers. I fear the lack of them

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that prints a string, followed by a new line, to `stdout` .

- Prototype: `void _puts(char *str);`

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

```
julien@ubuntu:~/0x05$ cat 3-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "I do not fear computers. I fear the lack of them - Isaac Asimov";
    _puts(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 3-ma
in.c 3-puts.c -o 3-puts
julien@ubuntu:~/0x05$ ./3-puts
I do not fear computers. I fear the lack of them - Isaac Asimov
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `3-puts.c`

☑ Done! | Help | Check your code | QA Review

4. I can only go one way. I've not got a reverse gear

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that prints a string, in reverse, followed by a new line.

- Prototype: `void print_rev(char *s);`

```
julien@ubuntu:~/0x05$ cat 4-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "I do not fear computers. I fear the lack of them - Isaac Asimov";
    print_rev(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 4-ma
in.c 4-print_rev.c -o 4-print_rev
julien@ubuntu:~/0x05$ ./4-print_rev
vomisA caasI - meht fo kcal eht raef I .sretupmoc raef ton od I
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `4-print_rev.c`

Done! | Help | Check your code | QA Review

5. A good engineer thinks in reverse and asks himself about the stylistic consequences of the components and systems he proposes

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that reverses a string.

- Prototype: `void rev_string(char *s);`

```
julien@ubuntu:~/0x05$ cat 5-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s[10] = "My School";

    printf("%s\n", s);
    rev_string(s);
    printf("%s\n", s);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-main.c 5-rev_
string.c -o 5-rev_string
julien@ubuntu:~/0x05$ ./5-rev_string
My School
loohcS yM
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `5-rev_string.c`

Done! | Help | Check your code | QA Review

6. Half the lies they tell about me aren't true

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that prints every other character of a string, starting with the first character, followed by a new line.

- Prototype: `void puts2(char *str);`

```
julien@ubuntu:~/0x05$ cat 6-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "0123456789";
    puts2(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 6-ma
in.c 6-puts2.c -o 6-puts2
julien@ubuntu:~/0x05$ ./6-puts2
02468
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `6-puts2.c`

Done! | Help | Check your code | QA Review

7. Winning is only half of it. Having fun is the other half

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that prints half of a string, followed by a new line.

- Prototype: `void puts_half(char *str);`
- The function should print the second half of the string
- If the number of characters is odd, the function should print the last `n` characters of the string, where `n = (length_of_the_string - 1) / 2`

```
julien@ubuntu:~/0x05$ cat 7-main.c
#include "main.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *str;

    str = "0123456789";
    puts_half(str);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putchar.c 7-ma
in.c 7-puts_half.c -o 7-puts_half
julien@ubuntu:~/0x05$ ./7-puts_half
56789
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `7-puts_half.c`

Done! | Help | Check your code | QA Review

8. Arrays are not pointers

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that prints `n` elements of an array of integers, followed by a new line.

- Prototype: `void print_array(int *a, int n);`
- where `n` is the number of elements of the array to be printed
- Numbers must be separated by comma, followed by a space
- The numbers should be displayed in the same order as they are stored in the array
- You are allowed to use `printf`

```
julien@ubuntu:~/0x05$ cat 8-main.c
#include "main.h"

/**
 * main - check the code for
 *
 * Return: Always 0.
 */
int main(void)
{
    int array[5];

    array[0] = 98;
    array[1] = 402;
    array[2] = -198;
    array[3] = 298;
    array[4] = -1024;
    print_array(array, 5);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 8-main.c 8-prin
t_array.c -o 8-print_array
julien@ubuntu:~/0x05$ ./8-print_array
98, 402, -198, 298, -1024
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `8-print_array.c`

☑ Done! | Help | Check your code | QA Review

9. strcpy

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

- Prototype: `char *_strcpy(char *dest, char *src);`

Write a function that copies the string pointed to by `src`, including the terminating null byte (`\0`), to the buffer pointed to by `dest`.

- Return value: the pointer to `dest`

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

```
julien@ubuntu:~/0x05$ cat 9-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char s1[98];
    char *ptr;

    ptr = _strcpy(s1, "First, solve the problem. Then, write the code\n");
    printf("%s", s1);
    printf("%s", ptr);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 9-main.c 9-strc
py.c -o 9-strcpy
julien@ubuntu:~/0x05$ ./9-strcpy
First, solve the problem. Then, write the code
First, solve the problem. Then, write the code
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `9-strcpy.c`

☑ Done! | Help | Check your code | QA Review

10. Great leaders are willing to sacrifice the numbers to save the people. Poor leaders sacrifice the people to save the numbers

#advanced

Score: 100.00% (*Checks completed: 100.00%*)

Write a function that convert a string to an integer.

- Prototype: `int _atoi(char *s);`
- The number in the string can be preceded by an infinite number of characters
- You need to take into account all the `-` and `+` signs before the number

- If there are no numbers in the string, the function must return `0`
- You are not allowed to use `long`
- You are not allowed to declare new variables of "type" array
- You are not allowed to hard-code special values
- We will use the `-fsanitize=signed-integer-overflow` gcc flag to compile your code.

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

```
julien@ubuntu:~/0x05$ cat 100-main.c
#include "main.h"
#include <stdio.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int nb;

    nb = _atoi("98");
    printf("%d\n", nb);
    nb = _atoi("-402");
    printf("%d\n", nb);
    nb = _atoi("          ------++++++-----+++++--98");
    printf("%d\n", nb);
    nb = _atoi("214748364");
    printf("%d\n", nb);
    nb = _atoi("0");
    printf("%d\n", nb);
    nb = _atoi("Suite 402");
    printf("%d\n", nb);
    nb = _atoi("         +      +    -    -98 Battery Street; San Francisco, CA 9411
1 - USA             ");
    printf("%d\n", nb);
    nb = _atoi("---++++ -++ Sui - te -   402 #cisfun :)");
    printf("%d\n", nb);
    return (0);
}
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 -fsanitize=sign
ed-integer-overflow 100-main.c 100-atoi.c -o 100-atoi
julien@ubuntu:~/0x05$ ./100-atoi
98
-402
-98
214748364
0
402
98
402
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `100-atoi.c`

Done! Help Check your code QA Review

11. Don't hate the hacker, hate the code

#advanced

Score: 100.00% (*Checks completed: 100.00%*)

Create a program that generates random valid passwords for the program 101-crackme (https://github.com/holbertonschool/0x04.c).

- You are allowed to use the standard library
- You don't have to pass the `betty-style` tests (you still need to pass the `betty-doc` tests)
- man `srand`, `rand`, `time`
- `gdb` and `objdump` can help

```
julien@ubuntu:~/0x05$ gcc -Wall -pedantic -Werror -Wextra 101-keygen.c -o 101-keygen
julien@ubuntu:~/0x05$ ./101-crackme "`./101-keygen`"
Tada! Congrats
julien@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x05-pointers_arrays_strings`
- File: `101-keygen.c`

Done! Help Check your code >_ Get a sandbox QA Review

Copyright © 2022 ALX, All rights reserved.