





# 0x0E. C - Structures, typedef


C

 By: Julien Barbier

 Weight: 1

 Project will start Aug 2, 2022 6:00 AM, must end by Aug 3, 2022 6:00 AM

 was released at Aug 2, 2022 12:00 PM

 An auto review will be launched at the deadline



## Resources



**Read or watch:**

(/)

- 0x0d. Structures.pdf (/rltoken/giS4eNQT2BQ9RLK0PMhgJQ)
- struct (C programming language) (/rltoken/MinJEDOHpeZs31qaXU8v1w)
- Documentation: structures (/rltoken/GiGFlaP5uUxeWLwfzdH15w)
- 0x0d. Typedef and structures.pdf (/rltoken/TGQ3RopVP7CjUTzF-XDXUw)
- typedef (/rltoken/aqqM2t7PLG5cyHaKwm5nBg)
- **Programming in C** by Stephen Kochan - *Chapter 8, Working with Structures p163-189*
- The Lost Art of C Structure Packing (/rltoken/emb4ohNT7XKi8Peep5lyeA) (*Advanced - not mandatory*)

## Learning Objectives

At the end of this project, you are expected to be able to explain to anyone

(/rltoken/qkcS8PT80wmgcNICEdBzrQ), **without the help of Google:**

### General

- What are structures, when, why and how to use them
- How to use typedef

### 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
- The only C standard library functions allowed are `printf` , `malloc` , `free` and `exit` .
- 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



- Don't forget to push your header file
- (/). All your header files should be include guarded



## Quiz questions

**Great!** You've completed the quiz successfully! Keep going! ([Hide quiz](#))

### Question #0

You should write documentation for all the structures you create

- ☒ True
- ☐ I'll do it 5 minutes before the deadline when I try Betty on my header file
- ☒ As soon as I write my structure.

### Question #1

The general syntax for a struct declaration in C is:

```
struct tag_name {  
    type member1;  
    type member2;  
    /* declare as many members as desired, but the entire structure size must be known to the compiler. */  
};
```

- ☒ True
- ☐ False
- ☐ Maybe

### Question #2

Those two codes do the same thing:

```
typedef struct point point;  
struct point {  
    int    x;  
    int    y;  
};  
point p = {1, 2};
```



```
typedef struct point point;  
(7) struct point {  
    int    x;  
    int    y;  
};  
point p = { .y = 2, .x = 1 };
```

- ☒ True
- ☐ False: the second does not compile
- ☐ False: the members of the structures will not have the same values

### Question #3

Given this code:

```
struct point {  
    int x;  
    int y;  
};  
struct point my_point = { 3, 7 };  
struct point *p = &my_point;
```

To set the member y of my variable my\_point to 98, I can do (select all valid answers):

- ☒ my\_point.y = 98;
- ☐ my\_point->y = 98;
- ☐ p.y = 98;
- ☒ (\*p).y = 98;
- ☒ p->y = 98;

## Tasks

### 0. Poppy

mandatory



( / )



Define a new type `struct dog` with the following elements:

- `name , type = char *`
- `age , type = float`
- `owner , type = char *`



```
julien@ubuntu:~/0x0d. structures, typedef$ cat 0-main.c
#include <stdio.h>
#include "dog.h"

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

    my_dog.name = "Poppy";
    my_dog.age = 3.5;
    my_dog.owner = "Bob";
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog.name, my_dog.age);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0
-main.c -o a
julien@ubuntu:~/0x0d. structures, typedef$ ./a
My name is Poppy, and I am 3.5 :) - Woof!
julien@ubuntu:~/0x0d. structures, typedef$
```

**Repo:**

- GitHub repository: alx-low\_level\_programming
- Directory: 0x0E-structures\_typedef
- File: dog.h

☒ Done![Help](#)[Check your code](#)**1. A dog is the only thing on earth that loves you more than you love yourself****mandatory**

Write a function that initialize a variable of type `struct dog`

- Prototype: `void init_dog(struct dog *d, char *name, float age, char *owner);`



```
julien@ubuntu:~/0x0d. structures, typedef$ cat 1-main.c
```

```
#include <stdio.h>
```

```
#include "dog.h"
```

```
/**
```

```
 * main - check the code
```

```
 *
```

```
 * Return: Always 0.
```

```
 */
```

```
int main(void)
```

```
{
```

```
    struct dog my_dog;
```

```
    init_dog(&my_dog, "Poppy", 3.5, "Bob");
```

```
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog.name, my_dog.age);
```

```
    return (0);
```

```
}
```

```
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1  
-main.c 1-init_dog.c -o b
```

```
julien@ubuntu:~/0x0d. structures, typedef$ ./b
```

```
My name is Poppy, and I am 3.5 :) - Woof!
```

```
julien@ubuntu:~/0x0d. structures, typedef$
```

### Repo:

- GitHub repository: alx-low\_level\_programming
- Directory: 0x0E-structures\_typedef
- File: 1-init\_dog.c

☒ Done!

**2. A dog will teach you unconditional love. If you can have that in your life, things won't be too bad**

**mandatory**







Write a function that prints a struct dog

- Prototype: `void print_dog(struct dog *d);`
- Format: see example bellow
- You are allowed to use the standard library
- If an element of `d` is `NULL`, print `(nil)` instead of this element. (if `name` is `NULL`, print `Name: (nil)`)
- If `d` is `NULL` print nothing.





```
julien@ubuntu:~/0x0d. structures, typedef$ cat 2-main.c
#include <stdio.h>
#include "dog.h"

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

    my_dog.name = "Poppy";
    my_dog.age = 3.5;
    my_dog.owner = "Bob";
    print_dog(&my_dog);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2
-main.c 2-print_dog.c -o c
julien@ubuntu:~/0x0d. structures, typedef$ ./c
Name: Poppy
Age: 3.500000
Owner: Bob
julien@ubuntu:~/0x0d. structures, typedef$
```

**Repo:**

- GitHub repository: alx-low\_level\_programming
- Directory: 0x0E-structures\_typedef
- File: 2-print\_dog.c

☒ Done![Help](#)[Check your code](#)

### 3. Outside of a dog, a book is a man's best friend. Inside of a dog it's too dark to read

**mandatory**



Define a new type `dog_t` as a new name for the type `struct dog`.

```
julien@ubuntu:~/0x0d. structures, typedef$ cat 3-main.c
#include <stdio.h>
#include "dog.h"

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

    my_dog.name = "Poppy";
    my_dog.age = 3.5;
    my_dog.owner = "Bob";
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog.name, my_dog.age);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3
-main.c -o d
julien@ubuntu:~/0x0d. structures, typedef$ ./d
My name is Poppy, and I am 3.5 :) - Woof!
julien@ubuntu:~/0x0d. structures, typedef$
```



**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0E-structures_typedef`
- File: `dog.h`

☒ Done!

Help

Check your code

#### 4. A door is what a dog is perpetually on the wrong side of

mandatory



Write a function that creates a new dog.

- Prototype: `dog_t *new_dog(char *name, float age, char *owner);`
- You have to store a copy of `name` and `owner`
- Return `NULL` if the function fails



```
julien@ubuntu:~/0x0d. structures, typedef$ cat 4-main.c
#include <stdio.h>
#include "dog.h"

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

    my_dog = new_dog("Poppy", 3.5, "Bob");
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog->name, my_dog->age);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4
-main.c 4-new_dog.c -o e
julien@ubuntu:~/0x0d. structures, typedef$ ./e
My name is Poppy, and I am 3.5 :) - Woof!
julien@ubuntu:~/0x0d. structures, typedef$
```

**Repo:**

- GitHub repository: alx-low\_level\_programming
- Directory: 0x0E-structures\_typedef
- File: 4-new\_dog.c

☒ Done![Help](#)[Check your code](#)

**5. How many legs does a dog have if you call his tail a leg? Four. Saying that a tail is a leg doesn't make it a leg**

**mandatory**



( / )



Write a function that frees dogs.

- Prototype: `void free_dog(dog_t *d);`



```

julien@ubuntu:~/0x0d. structures, typedef$ cat 5-main.c
#include <stdio.h>
#include "dog.h"

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

    my_dog = new_dog("Poppy", 3.5, "Bob");
    printf("My name is %s, and I am %.1f :) - Woof!\n", my_dog->name, my_dog->age);
    free_dog(my_dog);
    return (0);
}
julien@ubuntu:~/0x0d. structures, typedef$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5
-main.c 5-free_dog.c 4-new_dog.c -o f
julien@ubuntu:~/0x0d. structures, typedef$ valgrind ./f
==22840== Memcheck, a memory error detector
==22840== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==22840== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==22840== Command: ./f
==22840==
My name is Poppy, and I am 3.5 :) - Woof!
==22840==
==22840== HEAP SUMMARY:
==22840==    in use at exit: 0 bytes in 0 blocks
==22840==   total heap usage: 4 allocs, 4 frees, 1,059 bytes allocated
==22840==
==22840== All heap blocks were freed -- no leaks are possible
==22840==
==22840== For counts of detected and suppressed errors, rerun with: -v
==22840== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
julien@ubuntu:~/0x0d. structures, typedef$

```

**Repo:**

- GitHub repository: alx-low\_level\_programming
- Directory: 0x0E-structures\_typedef
- File: 5-free\_dog.c

☒ Done!






(/)

