# 0x10. C - Variadic functions

C

By: Julien Barbier

Weight: 1

Project over - took place from Oct 13, 2022 6:00 AM to Oct 14, 2022 6:00 AM

An auto review will be launched at the deadline

### In a nutshell…

- **Auto QA review:** 33.0/33 mandatory
- **Altogether: 100.0%**
  - Mandatory: 100.0%
  - Optional: no optional tasks

## Resources

**Read or watch**:

- stdarg.h (/rltoken/wLRJdO8pA2-Vb-rF2Y71sA)
- Variadic Functions (/rltoken/3gW8GycmyjarbJR76FkrzA)
- Const Keyword (/rltoken/_RRPCY32VODyN_r2HIEnBQ)

**man or help**:

- `stdarg`

## Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/ibS4gLVrbsqSDUdPhq_4Vg), **without the help of Google**:

### General

- What are variadic functions
- How to use `va_start`, `va_arg` and `va_end` macros

- Why and how to use the `const` type qualifier

(/)

## 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 `malloc`, `free` and `exit`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc... is forbidden
- You are allowed to use the following macros: `va_start`, `va_arg` and `va_end`
- 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 `variadic_functions.h`
- Don't forget to push your header file
- All your header files should be include guarded

# Tasks

### 0. Beauty is variable, ugliness is constant

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

(/)

Write a function that returns the sum of all its parameters.

- Prototype: `int sum_them_all(const unsigned int n, ...);`
- If `n == 0`, return `0`

```
julien@ubuntu:~/0x0f. variadic functions$ cat 0-main.c
#include <stdio.h>
#include "variadic_functions.h"

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

    sum = sum_them_all(2, 98, 1024);
    printf("%d\n", sum);
    sum = sum_them_all(4, 98, 1024, 402, -1024);
    printf("%d\n", sum);
    return (0);
}
julien@ubuntu:~/0x0f. variadic functions$ gcc -Wall -pedantic -Werror -Wextra -std=g
nu89 0-main.c 0-sum_them_all.c -o a
julien@ubuntu:~/0x0f. variadic functions$ ./a
1122
500
julien@ubuntu:~/0x0f. variadic functions$
```

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `0-sum_them_all.c`

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

## 1. To be is to be the value of a variable

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints numbers, followed by a new line.

- Prototype: `void print_numbers(const char *separator, const unsigned int n, ...);`
- where `separator` is the string to be printed between numbers
- and `n` is the number of integers passed to the function
- You are allowed to use `printf`
- If `separator` is `NULL`, don't print it
- Print a new line at the end of your function

```
julien@ubuntu:~/0x0f. variadic functions$ cat 1-main.c
#include "variadic_functions.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_numbers(", ", 4, 0, 98, -1024, 402);
    return (0);
}
julien@ubuntu:~/0x0f. variadic functions$ gcc -Wall -pedantic -Werror -Wextra -std=g
nu89 1-main.c 1-print_numbers.c -o b
julien@ubuntu:~/0x0f. variadic functions$ ./b
0, 98, -1024, 402
julien@ubuntu:~/0x0f. variadic functions$
```

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `1-print_numbers.c`

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

## 2. One woman's constant is another woman's variable

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints strings, followed by a new line.

- Prototype: `void print_strings(const char *separator, const unsigned int n, ...);`
- where `separator` is the string to be printed between the strings
- and `n` is the number of strings passed to the function
- You are allowed to use `printf`
- If separator is NULL, don't print it
- If one of the string is NULL, print `(nil)` instead

- Print a new line at the end of your function

(/)

```
julien@ubuntu:~/0x0f. Variadic functions$ cat 2-main.c
#include "variadic_functions.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_strings(", ", 2, "Jay", "Django");
    return (0);
}
julien@ubuntu:~/0x0f. Variadic functions$ gcc -Wall -pedantic -Werror -Wextra -std=g
nu89 2-main.c 2-print_strings.c -o c
julien@ubuntu:~/0x0f. Variadic functions$ ./c
Jay, Django
julien@ubuntu:~/0x0f. Variadic functions$
```

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `2-print_strings.c`

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

## 3. To be is a to be the value of a variable

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints anything.

- Prototype: `void print_all(const char * const format, ...);`
- where `format` is a list of types of arguments passed to the function
  - `c`: `char`
  - `i`: `integer`
  - `f`: `float`
  - `s`: `char *` (if the string is NULL, print `(nil)` instead
  - any other char should be ignored
  - see example
- You are not allowed to use `for`, `goto`, ternary operator, `else`, `do ... while`
- You can use a maximum of
  - 2 `while` loops

    - 2 `if`
- You can declare a maximum of `9` variables
- You are allowed to use `printf`
- Print a new line at the end of your function

```
julien@ubuntu:~/0x0f. Variadic functions$ cat 3-main.c
#include "variadic_functions.h"

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    print_all("ceis", 'B', 3, "stSchool");
    return (0);
}
julien@ubuntu:~/0x0f. Variadic functions$ gcc -Wall -pedantic -Werror -Wextra -std=g
nu89 3-main.c 3-print_all.c -o d
julien@ubuntu:~/0x0f. Variadic functions$ ./d
B, 3, stSchool
julien@ubuntu:~/0x0f. Variadic functions$
```

**Repo:**

- GitHub repository: `alx-low_level_programming`
- Directory: `0x10-variadic_functions`
- File: `3-print_all.c`

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