EPITECH.
PARIS GRADUATE SCHOOL OF DIGITAL INNOVATION

# my_printf

## Kick-off

## B1 - Unix System Programming

B-PSU-100

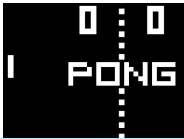
# Arguments

Up until now, we have used a number of fixed arguments:

```
void    my_putchar(char c);
int     my_example_function(char c, int i, char* s);
```

A function that takes:

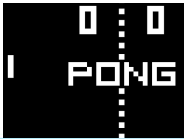
- 0 argument is called **niladic** function.
- 1 argument only is called **monadic** function.
- 2 arguments is called **dyadic** function.
- 3 arguments it called **triadic** function.
- multiple arguments is called **polyadic** function.



# Variadic functions

Sometimes, it is not possible to determine the number of arguments.
We need to use a variable number of arguments.
The function is then called **variadic**.

- **Advantage:**
  The number of arguments doesn't need to be known beforehand
- **Disadvantage:**
  This list doesn't contain the received types of parameters



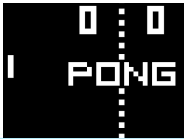
# Usage

- the first argument must be defined
- the argument list is represented by an ellipses: ...
- for example: printf

```
int     printf(char *str,...);
```

The first arguments has a char * type.
There are 1 to *n* arguments.

```
printf("Hello %s, you have %d points.\n", "Zaphod", 42);
```



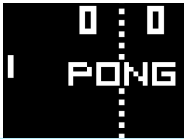
# stdarg

stdarg enables you to retrieve an argument list.
It uses the first string to get information about the following parameters.

man stdarg

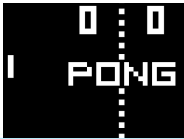
```
printf("Hello %s, you have %d points.\n", "Zaphod", 42);
```

%s => the first argument has a char * type.
%d => the second argument has a int type.



# Demonstration

Live coding demo



# Any questions

?