0x0D. C - Preprocessor

C

- By: Julien Barbier & Johan Euphrosine, Software Engineer at Google
- Weight: 1
- Project over - took place from Apr 11, 2022 6:00 AM to Apr 12, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 38.0/38 mandatory
- **Altogether: 100.0%**
    - Mandatory: 100.0%
    - Optional: no optional tasks

Resources

Read or watch:

- Understanding C program Compilation Process
- Object-like Macros
- Macro Arguments
- Pre Processor Directives in C
- The C Preprocessor
- Standard Predefined Macros
- include guard
- Common Predefined Macros

Learning Objectives

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

General

- What are macros and how to use them
- What are the most common predefined macros
- How to include guard your header files

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 and 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 _putchar
- 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
- 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! (Show quiz)

Tasks

0. Object-like Macro

mandatory

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

Create a header file that defines a macro named `SIZE` as an abbreviation for the token `1024`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 0-main.c
#include "0-object_like_macro.h"
#include "0-object_like_macro.h"
#include <stdio.h>

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

    s = 98 + SIZE;
    printf("%d\n", s);
    return (0);
}
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 0-main.c -o a
julien@ubuntu:~/0x0c. macro, structures$ ./a
1122
julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `0-object_like_macro.h`

Done! Help Check your code QA Review

1. Pi

mandatory

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

Create a header file that defines a macro named `PI` as an abbreviation for the token `3.14159265359`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 1-main.c
#include "1-pi.h"
#include "1-pi.h"
#include <stdio.h>

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

    r = 98;
    a = PI * r * r;
    printf("%.3f\n", a);
    return (0);
}
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 1-main.c -o b
julien@ubuntu:~/0x0c. macro, structures$ ./b
30171.855
julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `1-pi.h`

Done! Help Check your code QA Review

2. File name

mandatory

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

Write a program that prints the name of the file it was compiled from, followed by a new line.

- You are allowed to use the standard library

```
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 2-main.c -o c

julien@ubuntu:~/0x0c. macro, structures$ ./c

2-main.c

julien@ubuntu:~/0x0c. macro, structures$ cp 2-main.c 02-main.c

julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 02-main.c -o cc

julien@ubuntu:~/0x0c. macro, structures$ ./cc

02-main.c

julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `2-main.c`

Done! Help Check your code Get a sandbox QA Review

3. Function-like macro

mandatory

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

Write a function-like macro `ABS(x)` that computes the absolute value of a number `x`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 3-main.c

#include <stdio.h>

#include "3-function_like_macro.h"

#include "3-function_like_macro.h"


/**

 * main - check the code

 *

 * Return: Always 0.

 */

int main(void)
```

```
{
    int i;
    int j;

    i = ABS(-98) * 10;
    j = ABS(98) * 10;
    printf("%d, %d\n", i, j);
    return (0);
}
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 3-main.c -o d
julien@ubuntu:~/0x0c. macro, structures$ ./d
980, 980
julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `3-function_like_macro.h`

Done! Help Check your code QA Review

4. SUM

mandatory

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

Write a function-like macro `SUM(x, y)` that computes the sum of the numbers `x` and `y`.

```
julien@ubuntu:~/0x0c. macro, structures$ cat 4-main.c
#include <stdio.h>
#include "4-sum.h"
#include "4-sum.h"

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

    s = SUM(98, 1024);
    printf("%d\n", s);
    return (0);
}
julien@ubuntu:~/0x0c. macro, structures$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-main.c -o e
julien@ubuntu:~/0x0c. macro, structures$ ./e
1122
julien@ubuntu:~/0x0c. macro, structures$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0D-preprocessor`
- File: `4-sum.h`

Done! Help Check your code QA Review