Aranich, Marcel - 251453
Quera, Clàudia - 231197
Villarino, Jorge - 231351
Solans, Arnau - 216530
Garcia, David - 251587
Prat, Ariadna - 251281

System Overview

The C Preprocessor is a command-line tool developed in C that facilitates code preprocessing before compilation. Executed from the command line, it supports various flags like -c (eliminate comments), -d (replace directives), -all (combined -c and -d), and -help (display man page). The system employs pattern matching using the PatternMatcher structure for fixed and dynamic patterns, handling tasks like #define, #ifdef, #include, and comments. The preprocess function manages the input code, updating the writing buffer, and dynamic pattern matching is applied to certain directives. Specialized functions handle specific tasks like #ifdef and #endif processing, includes, defines and comments. The preprocessed content is saved in a new file with "_pp" appended before the extension, providing warnings about potential overwrites.

Detailed System Design

The main function, after parsing command-line arguments and initializing necessary variables, tit calls the preprocess function. The preprocess function is responsible for reading the content of the input C code file, detecting and handling various directives (such as #define, #include, etc.), and generating a preprocessed version of the code. The main function then manages the final steps, such as creating a new file with the preprocessed content and handling potential errors. The program ends with a return value of 0 for successful execution and 1 for any errors encountered.

We have divided our code into the following modules:

- **handle_includes:** When an include is found in the reading buffer, the corresponding handle_include function is evoked. There is a function for the compiler includes (i.e. “#include <stdlib.h>”) and another one for the files of your project (i.e. “#include “myheader.h””). These methods essentially read all the content of the file they have to include and use the preprocess() function in main.c in order to preprocess all the contents of the readed file recursively. Finally the proprocessed include is returned.

- **handle_constants:** The handle_constants function is in charge of when an identifier of a constant is detected, returning the proper value it equals to.

- **handle_macros:** When we detect a pattern that matches the one in the definition, we call handle_macro(). It identifies the macro's identifier, parameters, and content, allocating memory dynamically to store this information. The parsed data is organized into a three-dimensional array named result, which is then returned. Then, the program looks for the saved identifiers within the code and substituted with the corresponding macro content and the parameters values.

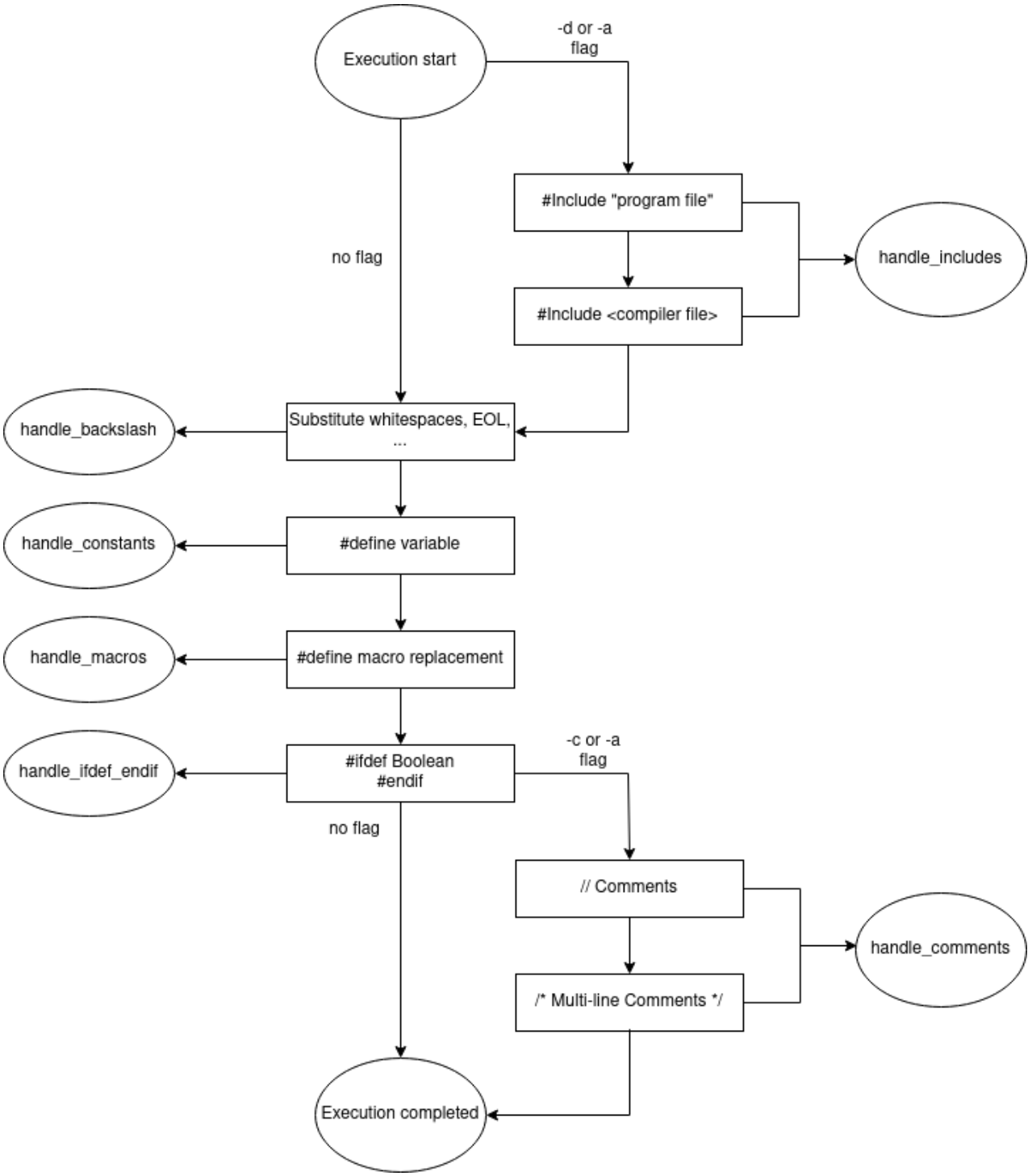
* We are considering joining handle_macros and handle_constants as a single task (handle defines) since handle_constant is a very simple case of handle_macros.

- **handle_comments:** The handle_comments module includes the handle_comments_simple and the handle_comments_multi functions, which will read the source code since the index passed as parameter until the end of the comment, and then will return the index at the end of the comment. Then, the main function can continue reading the source code from the index, skipping the comment. The first function will handle the single line comments and the second the multi line ones.

- **handle_ifdef_endif:** The function needs to return the contents of the ifdef if it has been never seen before or the empty string if it has been already seen. We need to use a data structure (i.e. the MultiString DS) in order to know which of the ifdefs have been inserted.

Module Interaction

We have included a workflow diagram to illustrate the interactions between the different modules in the program:



Individual Responsibilities

Breakdown of the tasks and roles assumed by each member of the team:

- **Handle includes:** Marcel Aranich and Arnau Solans
- **Handle ifdef and endif:** Ariadna Prat
- **Handle defines (constants and macros):** David Garcia and Clàudia Quera
- **Handle Comments:** Jorge Villarino