## Parse

The parse() function in the program is responsible for analysing each line individually. It operates by parsing the current line into fields, which are then stored in a data structure called **'field'** This structure consists of a **'char'** array with dimensions of 10 rows and 7 columns. This implies that every line can contain a maximum of 10 fields, with each field capable of accommodating up to 6 characters, with the final character being the null terminator. Each line is broken down into distinct elements, separated by whitespace or symbols. The parsing mechanism ignores any macro definitions found at the beginning of the program file and focuses solely on the lines following the 'PROG' line which indicates the start of the program. It is worth mentioning that the **'field'** data structure is defined globally, allowing other functions that require access to the **'field'** to utilize its contents. Furthermore, it is important to note that the 'field' data structure is reset to an empty state each time the parse() function is called. This deliberate clearing of the 'field' ensures that any previous data or remnants from previous parsing operations are completely removed, creating a fresh starting point for each new parsing operation. The function operates with only the filename as input and is responsible for determining the current line it is processing independently. It accomplishes this by utilizing a global variable named 'current_line'. Furthermore, a helper function called 'lineTrimmer' is employed to remove any unnecessary whitespace characters present in the lines before parsing them.

The function initiates by opening the specified file when it is executed. Subsequently, it proceeds to skip lines until it encounters the 'PROG' line, which signifies the start of the program. Once the 'PROG' line is located, the function parse commences to determine the current line. Once the current line is identified, the parsing operation begins. Initially, the line is tokenized into individual tokens based on specific delimiters. This process is accomplished by using the 'strtok' function from the string library. After tokenizing the line, the parsing operation proceeds to examine the first field to determine if it is a macro. This check is crucial for identifying and handling any macro-related components present in the line. Additionally, the parsing operation distinguishes between a normal macro and a special "if" macro during this examination. By differentiating between these types of macros, the parsing operation can appropriately process the field. After examining and processing the first field, the parsing operation proceeds to parse the remaining fields in a similar manner, ensuring that each field is handled appropriately accordingly. After concluding the parsing operation, the function completes its execution successfully by closing the file.