Description of the regular expressions parser

The program reads from file “input.txt” and writes to “output.txt”.
Regular expression is parsed sequentially. All transitions are stored in class NFA in a list. Transitions are of class type “Trans” which keeps vertex from, vertex to and transition symbol. An automaton is build in the following way. All symbols are being read while global counter of symbol in the regular expression is less then length of the regular expression itself. If symbol doesn't belongs to special symbols then automata of one symbol is build and put to stack. If two previous symbols we're not special and current symbol isn't special, then these automata are merged into one. If the symbol is “*” then kleene star is made with the previous operand. The two global functions do union and kleene automata: or_selection() and kleene(). If current symbol is “(“ or “|” the function parse_inside() is called again.
Testing is done as the following. Firstly, state 0 is appended to the list of traces, then for each symbol in the test string is searched the possible transition from the previous state with the current symbol or an epsilon transition (here I have “^” sign for this). If this transition exists then searche_trace() is called again passing either full copy of the test string or test string without the first symbol and full copy of trace. This function is based on a DFS algorithm. It stops whether current symbol is “\n” at the end of input line or if the state we're searching is the same as last recorded state and it is not a final one. Once solution trace if found, flag is raised, and program returns trace immediately. Also, when it sees symbol of string end, there can also be some epsilon transition after it, so it calls function do_last_epsilon_transaction() to do this. The last symbol of the file is deleted manually to keep the number of strings the same as the number of test cases.