# Compilers Project

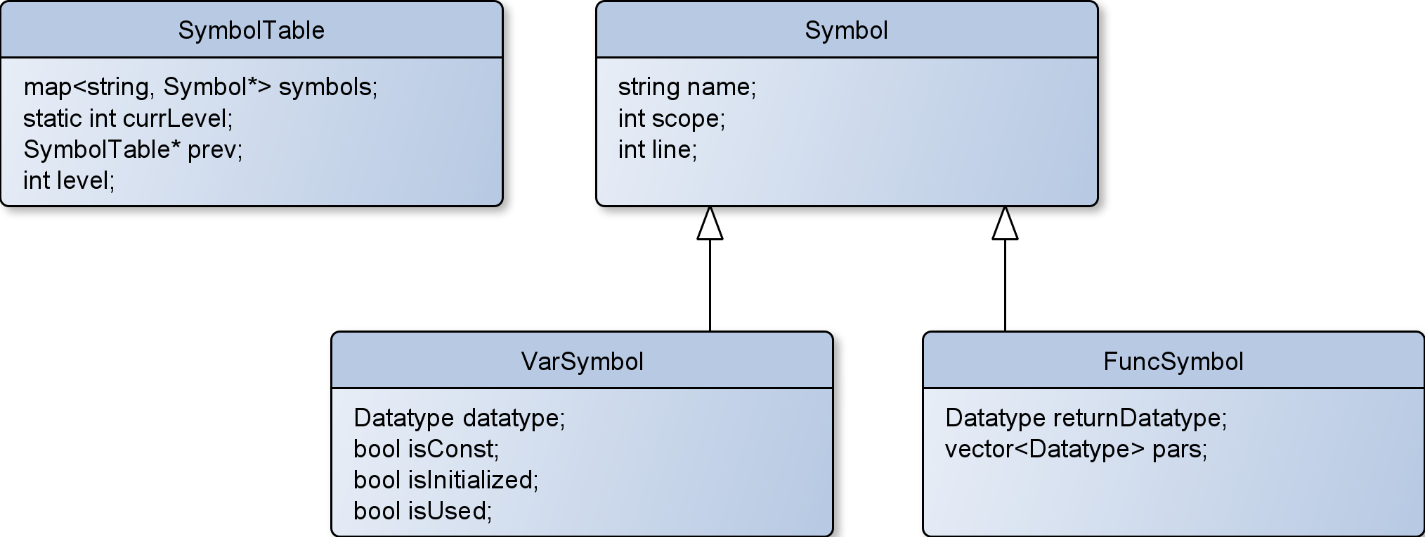
## Project overview

A simplified compiler for C language 89 which is known as ANSI C. The project consists of different parts: build system, lexer, parser, symbol table, parse and abstract syntax trees.

The lexer mainly contains the tokens and their regular expressions. The parser contains the context free grammar for the language besides the associated actions like dealing with the symbol table and constructing the PST (parse syntax tree) and AST (abstract syntax tree). The build system is responsible to keep track of the modified files to recompile and link.

### Symbol Table

The symbol table data structure consists of a map data structure for the symbols. STL map is used for that. It also contains a pointer to the containing table. If it’s the first symbol table, this pointer is initialized with NULL pointer. It has a level which indicates a specific scope. Meanwhile, the symbol data structure consists of the name of the symbol, the scope which is similar to the table level and finally the line where this symbol is declared in the input file. There’re two inheritances for the symbol: one for the variable symbol and another one for the function symbols.



### Code Generation

Polymorphism is used to generate the output quads instead of having many nested switches in the code. This made the code much modular and cleaner. Every node in the AST has its own “generate” function which outputs specific quads to the output file.

The generate function is called on the root of the AST, it outputs its specific quads then call its children generate function. The root treats the children as normal nodes and the polymorphism will take care of dynamic binding of the right generate functions of the children.

## Tools

The compiler is made using the GNU tools like flex, bison, gcc and Make. The programming language used to build the compiler is C++ besides its STL and OOP. For the GUI, python3 and PyQt5 are used. For graph generation, Graphviz is used. The lexer is made using flex. The parser is made using bison. The build system is made using GNU Make.

## Tokens and their description

For the tokens and their description, please visit this website: <https://docs.microsoft.com/en-us/cpp/c-language/c-tokens?view=msvc-170>

## Qudruples

|  |  |
| --- | --- |
| Quadruple | Description |
| .WORD X | Declare X |
| LD x Y | Y = x |
| ADD x y Z | Z = x + y |
| BITWISE\_OR x y Z | Z = x | y |
| BITWISE\_AND x y Z | Z = x & y |
| BITWISE\_XOR x y Z | Z = x ^ y |
| BITWISE\_NOT x Z | Z = ~ x |
| GR x y Z | Z = x > y |
| LR x y Z | Z = x < y |
| GRE x y Z | Z = x >= y |
| LRE x y Z | Z = x <= y |
| NOT x Z | Z = ! x |
| EQ x y Z | Z = x == y |
| NEQ x y Z | Z = x != y |
| AND x y Z | Z = x && y |
| OR x y Z | Z = x || y |
| ADD x y Z | Z = x + y |
| SUB x y Z | Z = x – y |
| MUL x y Z | Z = x \* y |
| DIV x y Z | Z = x / y |
| JMP Lx | Unconditional jump to label Lx |
| JMPF Tx Lx | Jump to Lx if Tx is falsy |
| JMPT Tx Lx | Jump to Lx if Tx is truthy |