

```
1: #ifndef SYMBOLTABLE_H
2: #define SYMBOLTABLE_H
3:
4: #include <string>
5: #include <vector>
6: #include <sstream>
7: #include <iomanip>
8: #include <stack>
9: #include "Lexer.h"
10: #include "Globals.h"
11:
12: // Begin instruction address at 1
13: static int instr_address = 1;
14:
15: class SymbolTable
16: {
17: public:
18:     struct Instr
19:     {
20:         int address;
21:         std::string op;
22:         int operand;
23:
24:         Instr(std::string op, int operand) : address(instr_address++), op(op), operand(operand) {}
25:     };
26:
27:     struct Symbol
28:     {
29:         Lexer::Token token;
30:         int address;
31:         std::string type;
32:
33:         Symbol(Lexer::Token t, int address, std::string type) : token(t), address(address), type(type) {}
34:     };
35:
36:     SymbolTable();
37:
38:     ~SymbolTable();
39:
40:     // Accessors
41:
42:         // Symbol Table
43:     int lookup(Lexer::Token t);
44:     std::string get_type(Lexer::Token token) const;
45:     int get_mem();
46:     int get_address(Lexer::Token token);
47:
48:         // Output
49:     std::string list();
50:     std::string list_instr();
51:
52:         // Instruction table
53:     int get_instr_address() const;
54:
55:         // Typestack
56:     std::string top_typestack() const;
57:     bool typestack_empty() const;
58:
59:     // Mutators
60:
61:         // Symbol Table
62:     bool insert(Lexer::Token t, std::string type);
63:     bool remove(Lexer::Token t);
64:
65:         // Instruction table
66:     void gen_instr(std::string op, int operand);
67:
68:         // Jumpstack
69:     void push_jumpstack(int address);
70:     void back_patch(int jump_addr);
71:
72:         // Typestack
73:     void push_typestack(std::string type);
74:     bool pop_typestack();
75:
76: private:
77:
78:     void incrementMem();
79:
80:     std::vector<Symbol> table;
81:     std::vector<Instr> instructions;
82:     std::vector<int> jumpstack;
83:     std::stack<std::string> typestack;
84:     int memaddress;
85:     std::ostringstream error;
86: };
87:
88: #endif // SYMBOLTABLE_H
```