College of Computer Studies

DEPARTMENT OF
SOFTWARE TECHNOLOGY

CSOPESY Seatwork
Instructor: Neil Patrick Del Gallego, Ph.D.

Use Calibri Font Size 11 for texts.

NAME:	**Daniel Gavrie Y. Clemente**
SECTION:	**S18**
DATE OF SEATWORK	**May 29, 2025**

SymbolTable.cpp

SymbolTable.cpp > Symbol > value

```cpp
#include <iostream>
#include <string>
#include <unordered_map>
#include <variant>
#include <memory>

// Enum for primitive C data types
enum class PrimitiveType {
    INT,
    CHAR,
    FLOAT,
    DOUBLE,
    SHORT,
    LONG,
    UNSIGNED_INT,
    SIGNED_INT
};


using PrimitiveValue = std::variant<int, char, float, double, short, long, unsigned int>;


struct Symbol {
    PrimitiveType type;
    PrimitiveValue value;
};

// SymbolTable class
class SymbolTable {
private:
    std::unordered_map<std::string, Symbol> table;

public:

    void store(const std::string& name, PrimitiveType type, PrimitiveValue value) {
        table[name] = {type, value};
    }


    bool update(const std::string& name, PrimitiveValue value) {
        if (table.find(name) != table.end()) {
            table[name].value = value;
            return true;
        }
        return false;
    }


    void print() const {
        for (const auto& [name, symbol] : table) {
            std::cout << "Variable: " << name << " = ";
            std::visit([](auto&& val) { std::cout << val; }, symbol.value);
            std::cout << std::endl;
        }
    }
};

int main() {
    // 1. Temporary symbol table
    SymbolTable tempTable;
    tempTable.store("myCSOPESYGrade", PrimitiveType::INT, 100);
    tempTable.print();

    // 2. Unique pointer to a symbol table
    std::unique_ptr<SymbolTable> uniqueTable = std::make_unique<SymbolTable>();
    uniqueTable->store("myCSOPESYGrade", PrimitiveType::INT, 100);
    uniqueTable->print();

    return 0;
}
```

The main() function shows two different ways to use the SymbolTable class. First, it creates a temporary symbol table directly on the stack and adds a variable called myCSOPESYGrade with a value of 100. Then, it creates another symbol

table on the heap, this time using a std::unique_ptr to handle memory management automatically. In both cases, the print() method is called to make sure the variable was successfully added.