Завдання

Модифікувати програму tab_symbol.cpp додатковою можливістю поміщати в таблицю символів

3. Знаки операцій порівняння і логічні оператори.

```
#include <iostream>
#include <unordered_map>
using namespace std;

class Node {
    string identifier, scope, type;
    int lineNo;
    Node* next;
public:
    Node() {
        next = nullptr;
    }
    Node(string key, string value, string type, int lineNo) {
        this->identifier = key;
        this->scope = value;
        this->type = type;
        this->lineNo = lineNo;
        next = nullptr;
    }
    void print() {
        cout << "Ім’я ідентифікатора: " << identifier
             << "\nТип: " << type
             << "\nОбласть видимості: " << scope
             << "\nНомер рядка: " << lineNo << endl;
    }
    friend class SymbolTable;
};

class SymbolTable {
    unordered_map<string, Node*> table;
public:
    SymbolTable() {}
    bool insert(string id, string scope, string type, int lineno) {
        if (table.find(id) != table.end()) {
            cout << id << " вже існує в таблиці.\n";
            return false;
        }
        Node* newNode = new Node(id, scope, type, lineno);
        table[id] = newNode;
        cout << id << " вставлений в таблицю.\n";
        return true;
    }
    void findAndPrintAll(string id) {
        bool found = false;
```

```
        for (auto& pair : table) {
            if (pair.first == id) {
                pair.second->print();
                found = true;
            }
        }
        if (!found) {
            cout << "Ідентифікатор " << id << " відсутній\n";
        }
    }
    bool deleteRecord(string id) {
        if (table.find(id) == table.end()) {
            return false;
        }
        delete table[id];
        table.erase(id);
        cout << id << " вилучений з таблиці.\n";
        return true;
    }
    bool modify(string id, string s, string t, int l) {
        if (table.find(id) == table.end()) {
            return false;
        }
        table[id]->scope = s;
        table[id]->type = t;
        table[id]->lineNo = l;
        cout << id << " модифікований.\n";
        return true;
    }
};

int main() {
    SymbolTable st;
    cout << "**** Таблиця символів ****\n";


    st.insert("if", "local", "keyword", 4);
    st.insert("number", "global", "variable", 2);
    st.insert("<", "global", "operator", 1);
    st.insert("=<", "global", "operator", 1);
    st.insert(">", "global", "operator", 1);
    st.insert(">=", "global", "operator", 1);
    st.insert("==", "global", "operator", 1);
    st.insert("&&", "global", "operator", 1);
    st.insert("||", "global", "operator", 1);

    cout << endl;
    // Пошук і друк всіх екземплярів 'if'
    cout << "Пошук і друк всіх екземплярів 'if':\n";
    st.findAndPrintAll("if");
    cout << endl;
```

```
    // Модифікація 'number'
    st.modify("number", "global", "variable", 3);
    cout << "\nПошук і друк всіх екземплярів 'number':\n";
    st.findAndPrintAll("number");
    cout << endl;

    // Пошук і друк всіх екземплярів '<'
    cout << "Пошук і друк всіх екземплярів '<':\n";
    st.findAndPrintAll("<");
    st.deleteRecord("<");
    cout << endl;

    // Пошук і друк всіх екземплярів '<='
    cout << "Пошук і друк всіх екземплярів '<=':\n";
    st.findAndPrintAll("<=");
    st.deleteRecord("<=");
    cout << endl;

     // Пошук і друк всіх екземплярів '>'
    cout << "Пошук і друк всіх екземплярів '>':\n";
    st.findAndPrintAll(">");
    st.deleteRecord(">");
    cout << endl;

    // Пошук і друк всіх екземплярів '>='
    cout << "Пошук і друк всіх екземплярів '>':\n";
    st.findAndPrintAll(">=");
    st.deleteRecord(">=");
    cout << endl;

    // Пошук і друк всіх екземплярів '=='
    cout << "Пошук і друк всіх екземплярів '==':\n";
    st.findAndPrintAll("==");
    st.deleteRecord("==");
    cout << endl;

    // Пошук і друк всіх екземплярів '&&'
    cout << "Пошук і друк всіх екземплярів '&&':\n";
    st.findAndPrintAll("&&");
    st.deleteRecord("&&");
    cout << endl;

    // Пошук і друк всіх екземплярів '||'
    cout << "Пошук і друк всіх екземплярів '||':\n";
    st.findAndPrintAll("||");
    st.deleteRecord("||");
    cout << endl;

    return 0;
}
```

```
**** Таблиця символів ****
if вставлений в таблицю.
number вставлений в таблицю.
< вставлений в таблицю.
=< встав?ений в таблицю.
> вставлений в таблицю.
>= вставлений в таблицю.
== вставлений в таблицю.
&& вставлений в таблицю.
|| вставлений в таблицю.

По?ук і друк всіх екземплярів 'if':
Ім'я ідентифікатора: if
Тип: keyword
Область видимості: local
Номер рядка: 4

number модифікований.

Пошук і друк всіх екземплярів 'number':
Ім'я ідентифікатора: number
Тип: variable
Область видим?сті: global
Номер рядка: 3

Пошук і друк всіх екземпляр?в '<':
Ім'я ідентифікатора: <
Тип: operator
Область видимості: global
Номер рядка: 1
< вилучений з таблиці.

Пошу?і друк всіх екземплярів '<=':
Ідентифікатор <= відсут?ій

Пошук і друк всіх екземплярів '>':
Ім'я ідентифікатора: >
```

```
Тип: operator
Область видимості: global
Номер ряд?а: 1
> вилучений з таблиці.

Пошук і друк всіх екземп?рів '>':
Ім'я ідентифікатора: >=
Тип: operator
Область в?димості: global
Номер рядка: 1
>= вилучений з таблиці.

?ошук і друк всіх екземплярів '==':
Ім'я ідентифікато?а: ==
Тип: operator
Область видимості: global
Номер рядка: 1
== вилучений з таблиці.

Пошук і друк всіх екземплярів '&&':
Ім'я ідентифікатора: &&
Тип: operator
Область види?ості: global
Номер рядка: 1
&& вилучений з таблиці.

Пош?к і друк всіх екземплярів '||':
Ім'я ідентифікатора: ||
Тип: operator
Область видимості: global
Номер рядка: 1
|| в?лучений з таблиці.
```