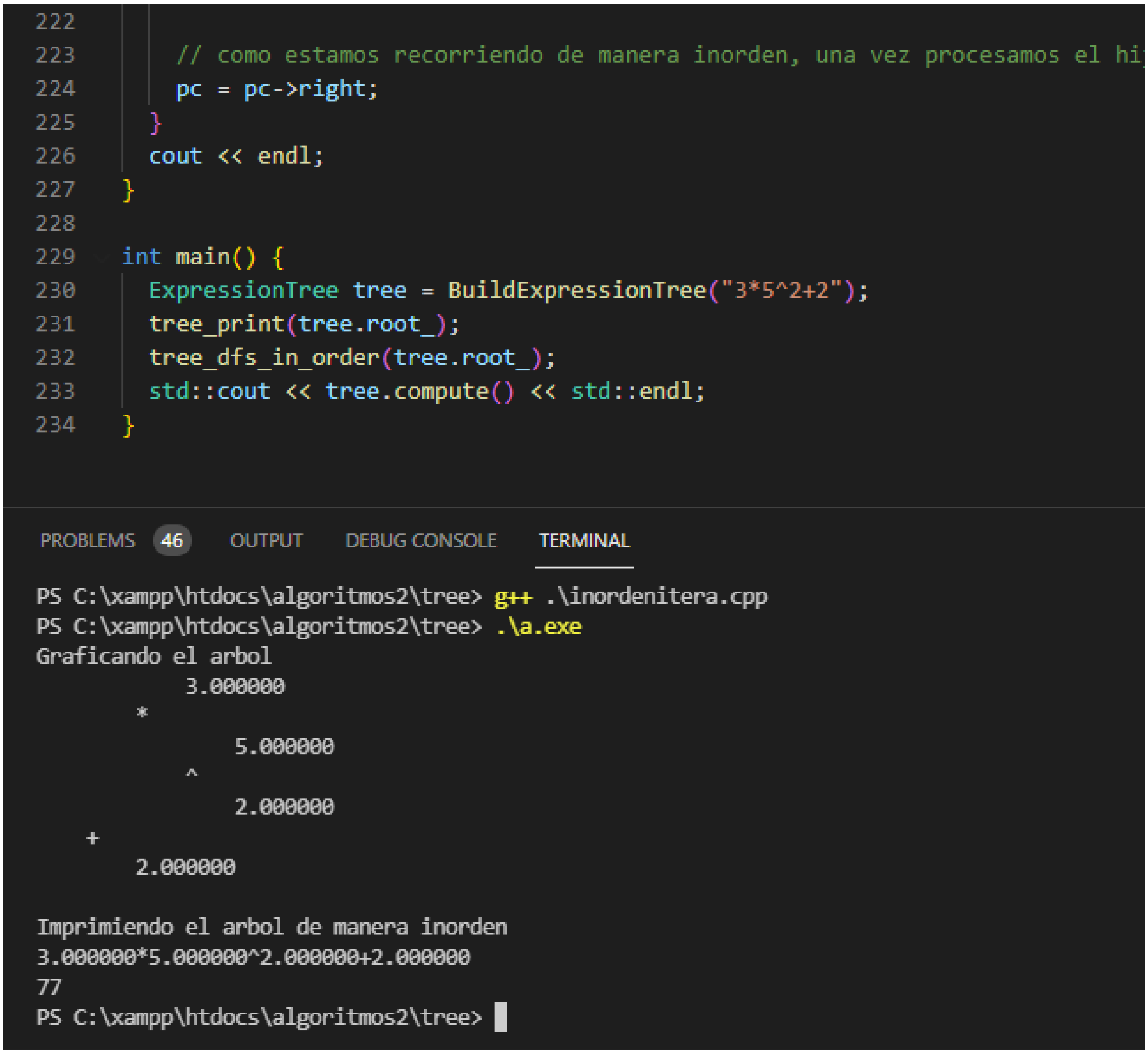
TRABAJO PRÁCTICO: Recorrido INORDEN ITERATIVO
Reynaldo Cusi Ascencio
Estructura de Datos

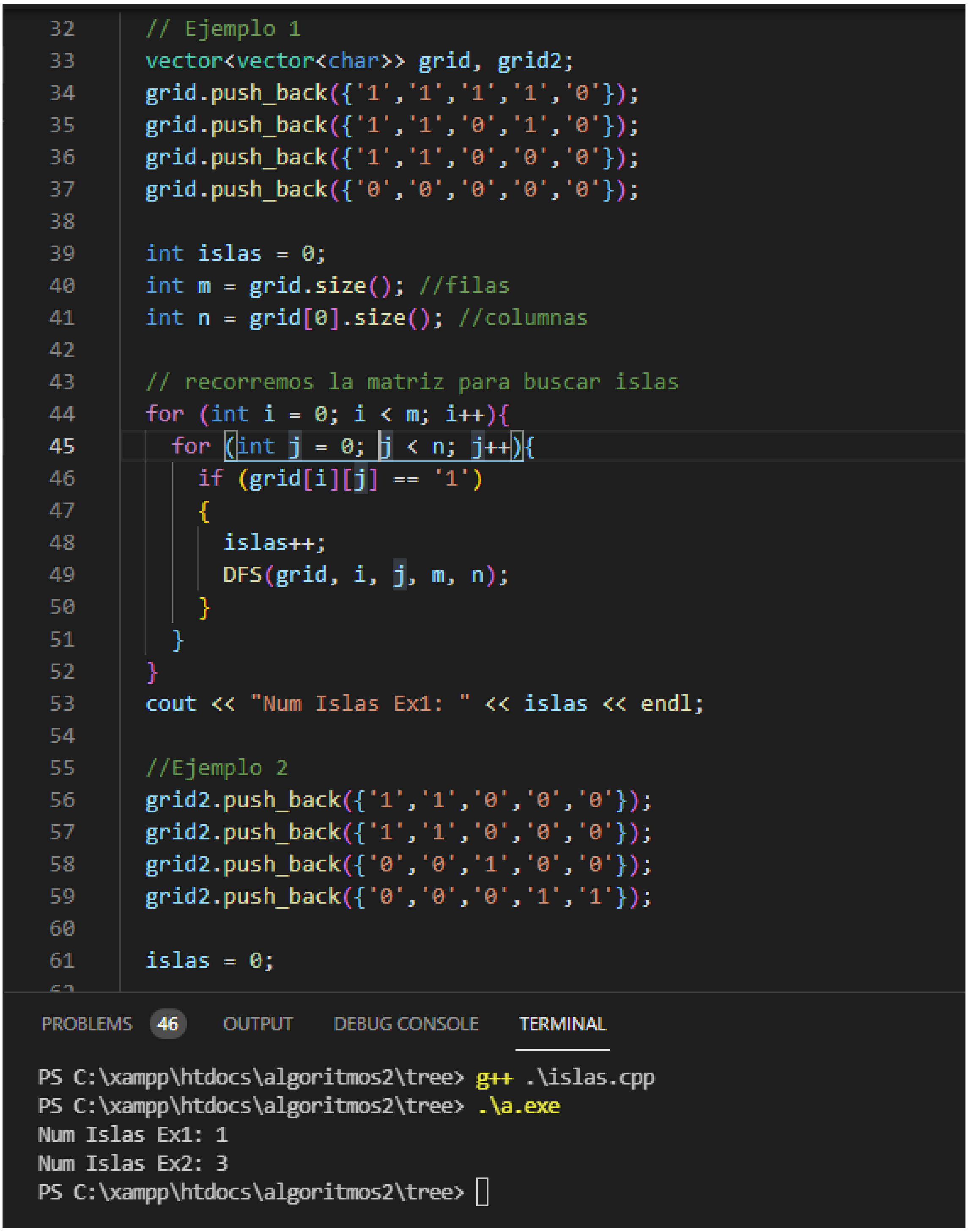
1. void tree_dfs_in_order(ExprNode *root)// iterativo con stack
2. implementar “tree_print” Iterativo con stacks
void tree_print(ExprNode *root, int level = 0);



```cpp

        // como estamos recorriendo de manera inorden, una vez procesamos el hi
        pc = pc->right;
    }
    cout << endl;
}

int main() {
    ExpressionTree tree = BuildExpressionTree("3*5^2+2");
    tree_print(tree.root_);
    tree_dfs_in_order(tree.root_);
    std::cout << tree.compute() << std::endl;
}
```

```
PS C:\xampp\htdocs\algoritmos2\tree> g++ .\inordenitera.cpp
PS C:\xampp\htdocs\algoritmos2\tree> .\a.exe
Graficando el arbol
            3.000000
        *
                5.000000
            ^
                2.000000
    +
        2.000000

Imprimiendo el arbol de manera inorden
3.000000*5.000000^2.000000+2.000000
77
PS C:\xampp\htdocs\algoritmos2\tree> 
```

3. aplicar dfs | bfs a:



```cpp
// Ejemplo 1
vector<vector<char>> grid, grid2;
grid.push_back({'1','1','1','1','0'});
grid.push_back({'1','1','0','1','0'});
grid.push_back({'1','1','0','0','0'});
grid.push_back({'0','0','0','0','0'});

int islas = 0;
int m = grid.size(); //filas
int n = grid[0].size(); //columnas

// recorremos la matriz para buscar islas
for (int i = 0; i < m; i++){
  for (int j = 0; j < n; j++){
    if (grid[i][j] == '1')
    {
      islas++;
      DFS(grid, i, j, m, n);
    }
  }
}
cout << "Num Islas Ex1: " << islas << endl;

//Ejemplo 2
grid2.push_back({'1','1','0','0','0'});
grid2.push_back({'1','1','0','0','0'});
grid2.push_back({'0','0','1','0','0'});
grid2.push_back({'0','0','0','1','1'});

islas = 0;
```

```
PS C:\xampp\htdocs\algoritmos2\tree> g++ .\islas.cpp
PS C:\xampp\htdocs\algoritmos2\tree> .\a.exe
Num Islas Ex1: 1
Num Islas Ex2: 3
PS C:\xampp\htdocs\algoritmos2\tree> 
```