INSPER

Felipe Frid Buniac

BOOLEAN PARENTHESIZATION

Implementação

São Paulo

2017

```
int countParenth(char symb[], char oper[], int n)
{
    int F[n][n], T[n][n];

    for (int i = 0; i < n; i++)
    {
        F[i][i] = (symb[i] == 'F')? 1: 0;
        T[i][i] = (symb[i] == 'T')? 1: 0;
    }
```

Iniciamos o código criando duas matrizes de programação dinâmica (F[n][n] e T[n][n]).

- Preenchemos a diagonal destas matrizes de acordo com a entrada passadas.
- Todas as entradas em T[i][i] são 1 se o símbolo [i] for True.
- Todas as entradas em F[i][i] são 1 se o símbolo [i] for False.
- n = Número de Operandos.

```
for (int gap=1; gap<n; ++gap)
{
    for (int i=0, j=gap; j<n; ++i, ++j)
    {
        T[i][j] = F[i][j] = 0;
        for (int g=0; g<gap; g++)
        {
```

Vamos preencher (T[i][i+1], T[i][i+2], T[i][i+3] e F[i][i+1], F[i][i+2], F[i][i+3]...)em ordem.

- gap = tamanho da distância entre operadores.
- i e j = definem posição da matriz.
- Vamos encontrar onde devemos colocar entre parênteses usando o valor atual de gap.

```
    for (int gap=1; gap<n; ++gap)
    {
        for (int i=0, j=gap; j<n; ++i, ++j)
        {
            T[i][j] = F[i][j] = 0;
            for (int g=0; g<gap; g++)
            {
                int k = i + g;

                int tik = T[i][k] + F[i][k];
                int tkj = T[k+1][j] + F[k+1][j];

                if (oper[k] == '&')
                {
                    T[i][j] += T[i][k]*T[k+1][j];
                    F[i][j] += (tik*tkj - T[i][k]*T[k+1][j]);
                }
                if (oper[k] == '|')
                {
                    F[i][j] += F[i][k]*F[k+1][j];
                    T[i][j] += (tik*tkj - F[i][k]*F[k+1][j]);
                }
                if (oper[k] == '^')
                {
                    T[i][j] += F[i][k]*T[k+1][j] + T[i][k]*F[k+1][j];
                    F[i][j] += T[i][k]*T[k+1][j] + F[i][k]*F[k+1][j];
                }
            }
        }
    }
    return T[0][n-1];
}
```

```
int k = i + g;
```

- k = local para colocar entre parênteses usando o valor do gap.

```
int tik = T[i][k] + F[i][k];
```

- tik = Armazena o Total[i][k]

```
int tkj = T[k+1][j] + F[k+1][j];
```

- tkj = Armazena o Total[k+1][j]

```
if (oper[k] == '&')
{
    T[i][j] += T[i][k]*T[k+1][j];
    F[i][j] += (tik*tkj - T[i][k]*T[k+1][j]);
}
```

- Fórmula recursiva do operador AND.

```
if (oper[k] == '|')
{
    F[i][j] += F[i][k]*F[k+1][j];
    T[i][j] += (tik*tkj - F[i][k]*F[k+1][j]);
}
```

- Fórmula recursiva do operador OR.

```
if (oper[k] == '^')
{
    T[i][j] += F[i][k]*T[k+1][j] + T[i][k]*F[k+1][j];
    F[i][j] += T[i][k]*T[k+1][j] + F[i][k]*F[k+1][j];
}
```

- Fórmula recursiva do operador OR.

```
return T[0][n-1];
```

- Retorna a posição da resposta da matriz dinâmica de True.

```
int main()
{
    char symbols[] = "TTFT";
    char operators[] = "|&^";
    int n = strlen(symbols);

    cout << countParenth(symbols, operators, n);
    return 0;
}
```

- Programa driver para testar o código acima.
- Recebe 2 vetores, 1 de símbolos e 1 de operadores.