La pile 📚

Les bases

- La pile est une structure de données qui permet seulement d'accéder au dernier élément ajouté, ou de le retirer.
- En cela, une pile est semblable à son équivalent dans le monde réel (pensez à une pile d'assiettes).
- On dit de cette structure de données qu'elle est de type *LIFO*, pour *Last In, First Out*. Autrement dit: dernier arrivé, premier sorti.

La pile de la STL

- La pile de la STL se nomme `stack`. Comme la classe `vector`, il s'agit d'une classe avec *template*.
- Les méthodes les plus importantes de la classe `stack` sont les suivantes:
    - `push` pour ajouter un élément sur le dessus de la pile;
    - `size` pour connaître le nombre d'éléments de la pile;
    - `empty` pour savoir si la pile est vide (retourne un booléen);
    - `top` pour accéder à l'élément sur le dessus de la pile;
    - `pop` pour retirer cet élément.

Exemple 1

```
#include <iostream>
#include <stack>

using namespace std;

void main() {
    stack<string> pile;

    // Ajouter des éléments à la pile
    pile.push("steak");
    pile.push("blé d'inde");
    pile.push("patates");

    cout << "La pile contient " << pile.size() << " éléments." << endl;

    // Afficher les éléments de la pile
    while (!pile.empty()) {
        cout << pile.top() << endl; // `top` retourne l'élément au sommet de la pile
        pile.pop(); // `pop` enlève l'élément au sommet de la pile
```

```
    }

    cout << "La pile contient " << pile.size() << " éléments." << endl;
}
```

Ce code produit l'affichage suivant:

```
La pile contient 3 éléments.
patates
blé d'inde
steak
La pile contient 0 éléments.
```

Exemple 2

Voici un programme qui utilise une pile pour vérifier si le mot entré par l'utilisateur est un palindrome (c'est-à-dire un mot qui se lit dans les deux sens, par exemple "kayak"):

```
stack<char> pile;
string mot;
string motInverse = "";

cout << "Entrer un mot: ";
cin >> mot;

for (int i = 0; i < mot.length(); i++) {
    pile.push(mot[i]);
}
while (!pile.empty()) {
    motInverse += pile.top();
    pile.pop();
}

if (mot == motInverse) {
    cout << "Le mot est un palindrome" << endl;
}
else {
    cout << "Le mot n'est pas un palindrome" << endl;
}
```