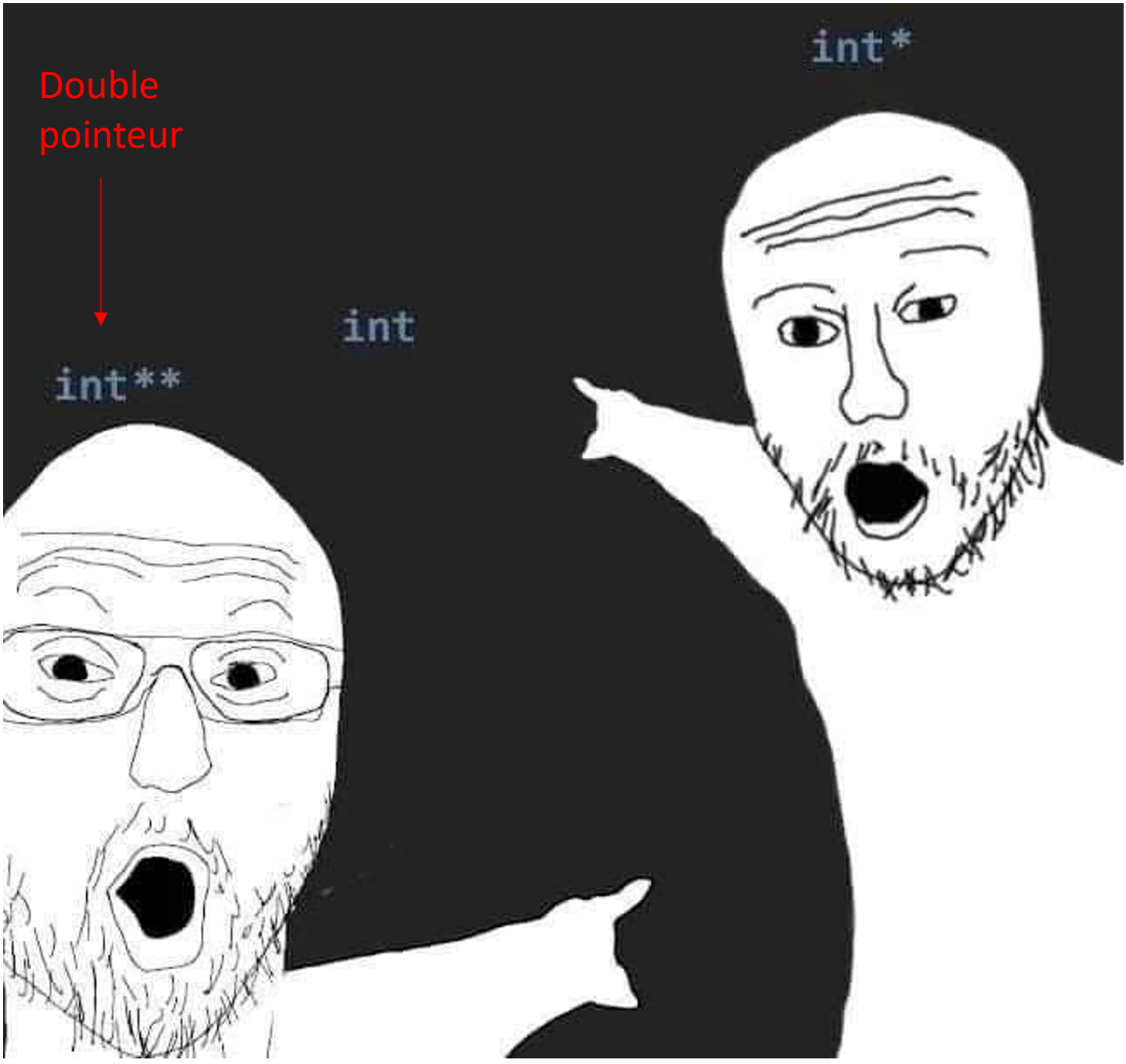
Les doubles pointeurs



Les bases

- **Rappel:** un pointeur contient l'adresse mémoire d'une donnée.
- Un pointeur de pointeur (ou double pointeur) contient donc... l'adresse d'une adresse!
- Les doubles pointeurs sont utiles pour créer des matrices dynamiques, puisqu'une matrice est un tableau de tableaux.
- Une matrice dynamique est donc un pointeur vers un tableau de pointeurs.

Exemples

Créer un double pointeur

```
int** matrice = nullptr;
```

Allouer la mémoire pour une matrice de 5 lignes par 10 colonnes

```
int nbLignes = 5;
int nbColonnes = 10;

// Ajouter 5 lignes à notre matrice
matrice = new int* [nbLignes];

// Ajouter 10 colonnes à chaque ligne
for (int i = 0; i < nbLignes; i++) {
    *(matrice + i) = new int[nbColonnes];
    // OU BIEN
    // matrice[i] = new int[10];
}
```

Initialiser les éléments de la matrice

```
for (int i = 0; i < nbLignes; i++) {
    for (int j = 0; j < nbColonnes; j++) {
        //*(*(matrice + i) + j) = (i + 1) * 10 + j;
        // OU BIEN
        matrice[i][j] = (i + 1) * 10 + j;
    }
}
```

Afficher les éléments de la matrice

```
for (int i = 0; i < nbLignes; i++) {
    for (int j = 0; j < nbColonnes; j++) {
        cout << *(*(matrice + i) + j) << " ";
    }
    cout << endl;
}
```

Cela affichera:

```
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
```

Désallouer la mémoire

```
for (int i = 0; i < nbLignes; i++) {
    delete[] *(matrice + i);
    // OU BIEN
    // delete[] matrice[i];
}
delete[] matrice;
matrice = nullptr;
```