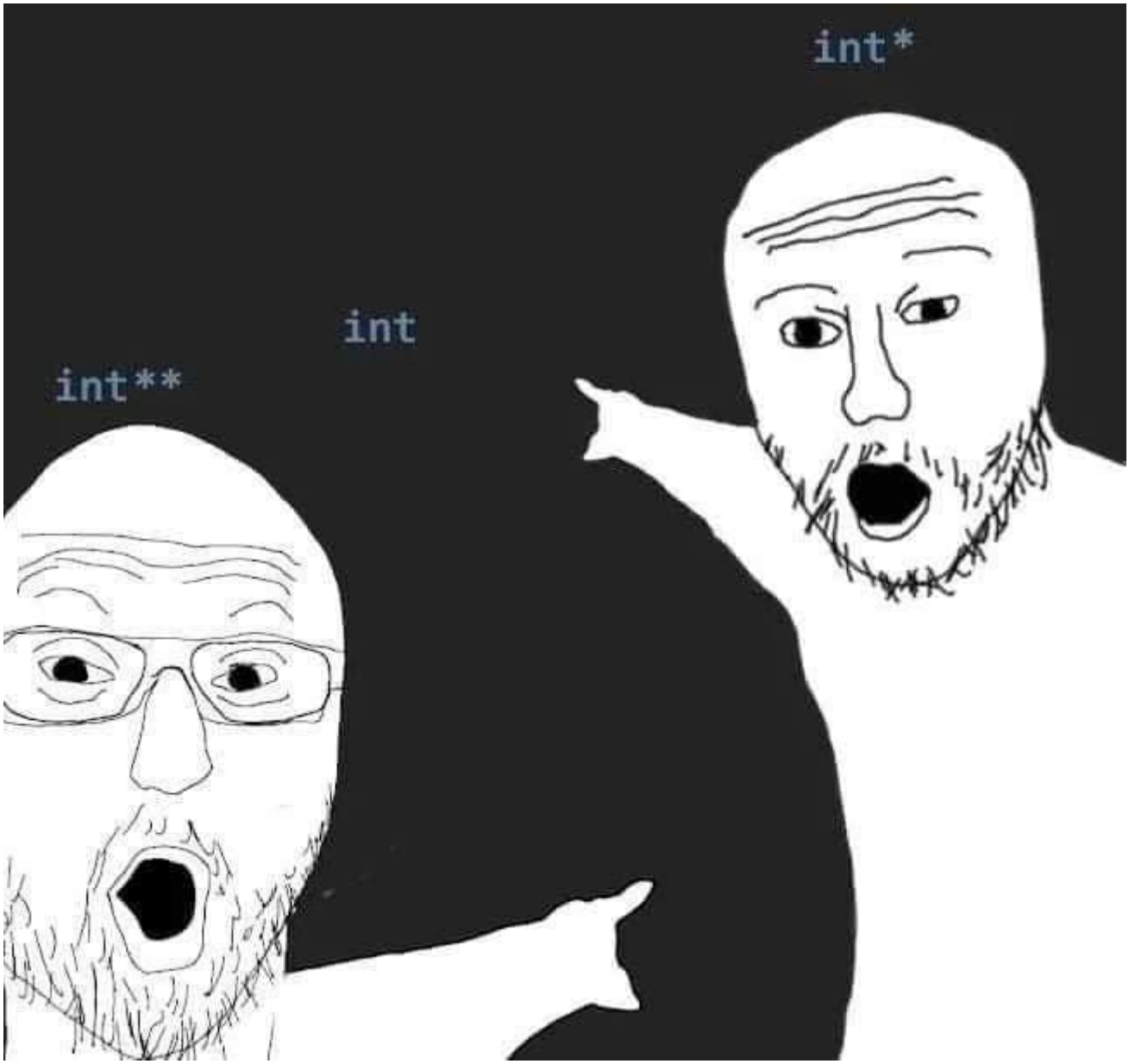
Les pointeurs



Les bases

- Toute variable possède une adresse mémoire.
- On peut obtenir l'adresse d'une variable à l'aide de l'opérateur `&`, ex:

```
int maVariable;
cout << &maVariable; // Affiche l'adresse de maVariable
```

- Un **pointeur** est une variable qui contient une adresse mémoire. Celle-ci peut pointer vers un type primitif, un objet ou un tableau.
- Avec les pointeurs, il faut allouer et désallouer soit-même la mémoire. Il s'agit donc d'**allocation dynamique**.

Utilisation

Pointeur de nombre entier (int)

```
// Déclarer un pointeur
int* pointeur;

/*
Le pointeur n'a pas été initialisé, donc présentement on ne sait pas vers quelle
adresse mémoire il pointe.
On aurait pu l'initaliser à nullptr (int* pointeur = nullptr;) pour s'assurer qu'il
ne pointe vers rien.
Dans un objet, on initialise les pointeurs dans le constructeur.
*/

// Créer un nombre entier et assigner son adresse au pointeur
pointeur = new int;

// Changer la valeur du nombre entier
*pointeur = 42;

// Afficher le nombre entier
cout << *pointeur;

// Afficher l'adresse du nombre entier
cout << pointeur;

// Désallouer la mémoire
delete pointeur;
```

Pointeur de tableau de nombres entiers (int[])

```
// Déclarer un pointeur
int* tableau = nullptr;

// Créer un tableau de 10 nombre entiers et assigner son adresse au pointeur
tableau = new int[10];

// Afficher le 3e élément du tableau
cout << *(tableau + 2);
// OU BIEN
```

```
cout << tableau[2];

// Assigner la valeur 42 au 3e élément du tableau
*(tableau + 2) = 42;
// OU BIEN
tableau[2] = 42;

// Détruire le tableau
delete[] tableau;

// Créer un nouveau tableau de 20 éléments
tableau = new int[20];

// Détruire le nouveau tableau
delete[] tableau;
```

Détecter une erreur lors de l'allocation de mémoire

```
int* tableau = nullptr;

try {
    tableau = new int[10];
} catch (bad_alloc& e) {
    cout << "Mémoire insuffisante pour l'allocation dynamique du tableau." << endl;
    exit(EXIT_FAILURE);
}
```