Débogguer un algorithme

- IDE et erreur de compilation
- Valeur test
- Trace papier
- Logger
- Débogger
- Canard

Erreur de compilation

- Les premiers type d'erreur, et les plus évidentes, sont les erreurs de syntaxes.
- L'IDE souligne généralement ces erreurs en rouge et mettre la souris par dessus fait afficher une description.
- Les plus fréquents sont:
  - L'oublie d'un ; à la fin d'une instruction
  - Un *typo* dans une nom de variable ou un nom de fonction
  - Un erreur de case (majuscule vs minuscule)
- Votre premier réflexe devrait être de repérer tout ce qui sort en rouge dans votre IDE et de le corriger.
- Une fois les erreurs de langages réglées, on peut s'attaquer aux erreurs d'algorithme.

Valeur test

- Afin de déterminer si l'algorithme fonctionne correctement, nous recommandons généralement d'avoir un jeu de données de test.
- Il s'agit d'une combinaison entrées/sorties que l'on sait exact.
- Nous pouvons alors valider si, avec nos entrées, le programme nous donnes les bonnes sorties. Si ce n'est pas le cas, il faut débogguer sont algorithme.
- Les méthodes ci-dessous peuvent vous aider.

Trace papier

- Lorsqu'on parle de trace, on parle de conserver un trace de la valeur de chaque variable à une ligne donnée.
- La trace papier est la forme la plus simple, on écrit sur un bout de papier la valeur des variables d'un algorithme en pseudo code. Cela nous permet de valider les étapes.
- Par exemple, quel est la valeur finale de valeur1 et valeur2 dans l'algorithme ci-dessous.

```
ENTIER valeur1 = 12;
ENTIER valeur2 = 6;

ENTIER valeur3 = valeurs 2;
valeur2 = valeur1;
valeur1 = valeur3;
```

Trace papier

```
ENTIER valeur1 = 12;
ENTIER valeur2 = 6;

ENTIER valeur3 = valeurs 2; // valeur3 -> 6
valeur2 = valeur1;          // valeur2 -> 12
valeur1 = valeur3;          // valeur1 -> 6
```

- La valeur finale de valeur1 est 6 et la valeur finale de valeur2 est 12. On peut donc voir que l'algorithme ci-haut sert à inverser les valeurs de 2 variables.

Logger

- Lorsque nous utilisons un langage de programmation plutôt que du pseudo-code, nous avons l'avantage de pouvoir afficher des valeurs dans la console.
- C'est la seconde technique de débogage. Nous utilisons la console pour inscrire les valeurs des variables afin de confirmer les comportements.
- Attention, cette méthode ajoute plusieurs ligne de code inutile au résultat final. N'oubliez pas de nettoyer votre code une fois que vous avez trouvez le problème.

```
int valeur1 = 12;
int valeur2 = 6;

int valeur3 = valeurs 2;
Console.WriteLine($"Valeur3={valeur3}");  // valeur3=6
valeur2 = valeur1;
Console.WriteLine($"valeur2={valeur2}");  // valeur2=12
valeur1 = valeur3;
Console.WriteLine($"valeur1={valeur1}");  // valeur1=6
```

Déboggeur

- La plupart des IDE ont un déboggeur intégré.
- Il vous suffit de cliquer à gauche du numéro de ligne pour faire apparaître un point rouge.
- Ce point rouge s'appelle un *breakpoint*.
- Vous pouvez ensuite exécuter votre programme en mode *Debug* (le bouton *play* avec une petit *bug* à côté)
- Lors de l'exécution, le programme s'arrêtera à votre *breakpoint* et vous pourrez voir les valeurs des variables à cette ligne.
- C'est ensuite possible de faire évoluer le programme ligne par ligne.

Canard

- Quand plus rien de marche, sortez votre Canard
  - [https://en.wikipedia.org/wiki/Rubber_duck_debugging]
- Les statistiques ont démontré que d'expliquer notre algorithme à quelqu'un aide à le clairifier. Puisque la personne qui écoute le programmeur n'a rien à dire, elle n'est là que pour permettre au programmeur d'expliquer son algorithme, elle n'a même pas besoin d'être faites de chair et de sang.
  - Un canard en plastique est l'oreille attentive toute désignée pour vous écouter.