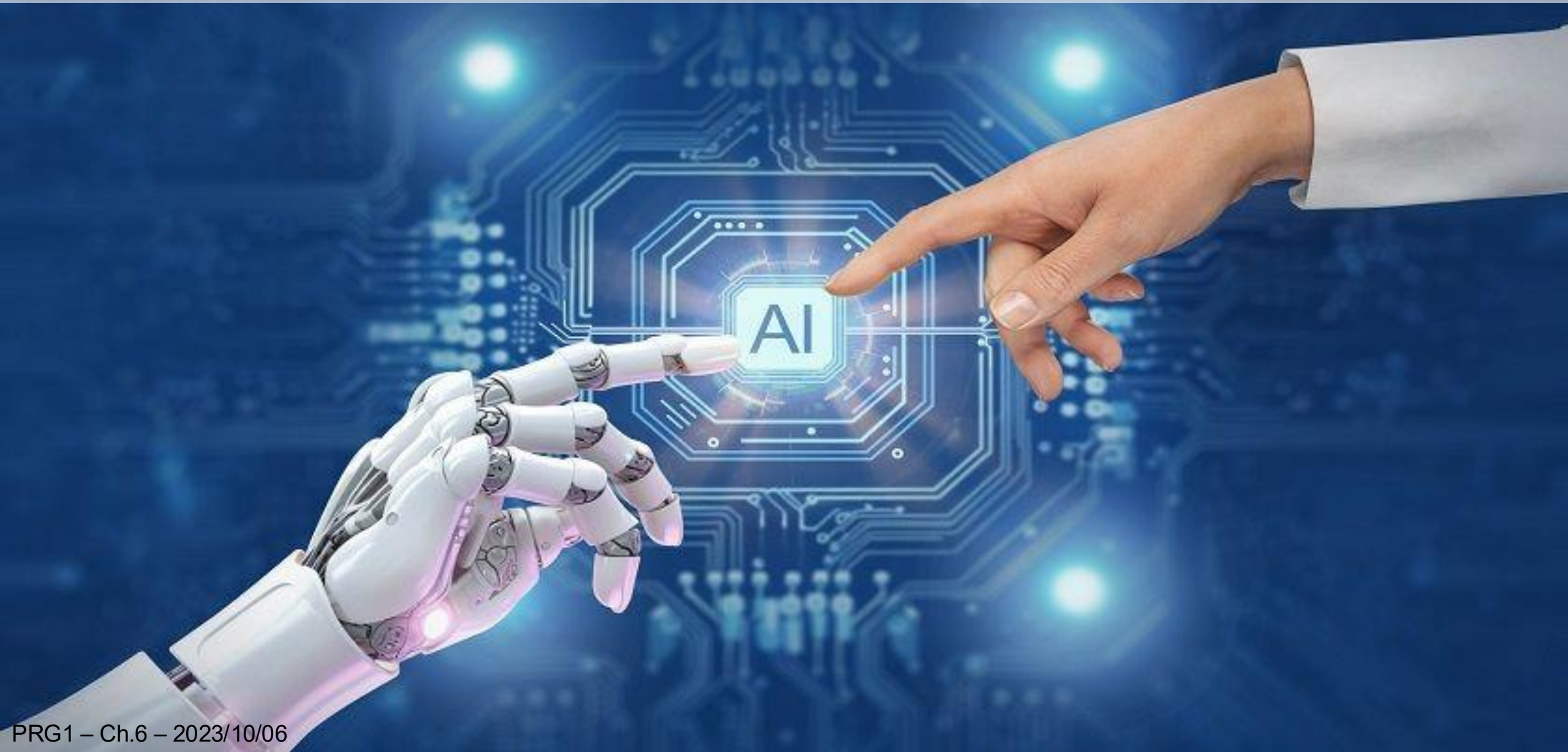


# Utilisation de l'AI en PRG1



# Principes



- Dans votre futur travail, vous utiliserez l'AI.
- Pour justifier votre salaire, il faudra que vous apportiez une plus value par rapport à quelqu'un qui utiliserait l'AI sans avoir été formé dans une HES.
- Pour cela, vous devez au minimum maîtriser les fondamentaux de la programmation, ce qui enrichira votre interaction avec l'AI et vous rendra plus efficace
- En PRG1, vous apprenez ces fondamentaux, pas à utiliser une AI
- En PRG1, vous serez évalués sur votre capacité à écrire des programmes sans AI, dans un environnement contrôlé pour que vous n'y ayez pas accès.
- A cause de l'ubiquité de l'AI aujourd'hui, votre travail à domicile ne sera pas évalué. Il vous sert uniquement à comprendre, apprendre, vous entraîner.

# Exemple : résoudre l'exercice 7.1.5 du recueil



Files

main

Go to file

> 01 - Introduction

> 02 - Elements de base

> 03 - Structures de Controle

> 04 - Fonctions

> 05 - Flux

> 06 - Types arithmétiques et conv...

> 07 - Structure\_Enum

01-01 - definition.md

01-02 - initialisation.md

01-03 - acces\_membres.md

01-04 - Date\_Personne.md

01-05 - est\_bissextile.md

01-06 - Nombre de jours entre ...

02-01 - Premiere et derniere let...

03-01 - Nombre de jours dans u...

PRG1\_Recueil\_Exercices / 07 - Structure\_Enum / 01-05 - est\_bissextile.md

gmbreguet

2023-11-07 17:43:09 - GMB

5be94da · 2 years ago

History

Preview

Code

Blame

32 lines (24 loc) · 722 Bytes

Raw

Download

Edit

## struct - fonction est\_bissextile

Ecrire une fonction *est\_bissextile* qui prend en paramètre une *Date* et retourne *true* si la date reçue en paramètre est en effet bissextile, *false* dans les cas contraires.

Une année est bissextile si :

- l'année est divisible par 400; ou
- l'année est divisible par 4 mais pas par 100

```
struct Date {
    uint8_t jour;
    uint8_t mois;
    uint16_t annee;
};
```

► Solutions

# Mauvaise approche



ChatGPT ▾

↑ Partager ...

OL

Peux tu me résoudre ce problème en C++ ? Ecrire une fonction `est_bissextile` qui prend en paramètre une `Date` et retourne `true` si la date reçue en paramètre est en effet bissextile, `false` dans les cas contraires.

Une année est bissextile si :

l'année est divisible par 400; ou

l'année est divisible par 4 mais pas par 100

```
struct Date {  
    uint8_t jour;  
    uint8_t mois;  
    uint16_t annee;  
};
```

# Mauvaise approche (2)



Voici une solution complète en C++ pour la fonction `est_bissextile` :

## ✓ Spécifications respectées :

- L'année est bissextile si elle est divisible par 400, ou divisible par 4 mais pas par 100.
- La fonction prend une structure `Date` en paramètre et retourne un booléen.

## ✓ Implémentation :

```
cpp                                                                    Copier Modifier

#include <cstdint> // pour uint8_t et uint16_t
#include <iostream>

struct Date {
    uint8_t jour;
    uint8_t mois;
    uint16_t annee;
};

bool est_bissextile(const Date& d) {
    uint16_t annee = d.annee;
    return (annee % 400 == 0) || ((annee % 4 == 0) && (annee % 100 != 0))
}
```

// Exemple d'utilisation

Copier Modifier

```
int main() {
    Date d1 = {1, 1, 2000}; // bissextile
    Date d2 = {1, 1, 1900}; // pas bissextile
    Date d3 = {1, 1, 2024}; // bissextile
    Date d4 = {1, 1, 2023}; // pas bissextile

    std::cout << "2000 : " << est_bissextile(d1) << "\n";
    std::cout << "1900 : " << est_bissextile(d2) << "\n";
    std::cout << "2024 : " << est_bissextile(d3) << "\n";
    std::cout << "2023 : " << est_bissextile(d4) << "\n";

    return 0;
}
```

## 🔍 Résultats attendus :

yaml

Copier Modifier

```
2000 : 1
1900 : 0
2024 : 1
2023 : 0
```

Souhaitez-tu également une fonction qui vérifie si **le jour existe dans le mois**, en tenant compte des années bissextiles (ex. 29 février. ↓)

# Meilleure approche



ChatGPT ▾

↑ Partager ...

Peux-tu m'écrire un programme de test pour cet exercice, sans m'en donner la solution ? Ecrire une fonction `est_bissextile` qui prend en paramètre une `Date` et retourne `true` si la date reçue en paramètre est en effet bissextile, `false` dans les cas contraires.

Une année est bissextile si :

l'année est divisible par 400; ou

l'année est divisible par 4 mais pas par 100

```
struct Date {  
    uint8_t jour;  
    uint8_t mois;  
    uint16_t annee;  
};
```





# Meilleure approche (2)



Voici un programme de test simple pour vérifier le bon fonctionnement d'une fonction

`est_bissextile(const Date& d)` que tu dois implémenter toi-même :

cpp

[Copier](#) [Modifier](#)

```
#include <iostream>
#include <cassert>

struct Date {
    uint8_t jour;
    uint8_t mois;
    uint16_t annee;
};

// Prototype de la fonction à implémenter
bool est_bissextile(const Date& d);

// Fonction de test
void tester_est_bissextile() {
    // Cas d'années bissextiles
    assert(est_bissextile({1, 1, 2000}) == true); // divisible par 400
    assert(est_bissextile({1, 1, 2024}) == true); // divisible par 4 ma
    assert(est_bissextile({29, 2, 1996}) == true); // autre année bissex
```

```
// Cas d'années non bissextiles
assert(est_bissextile({1, 1, 1900}) == false); // divisible par 100
assert(est_bissextile({1, 1, 2023}) == false); // non divisible par
assert(est_bissextile({28, 2, 2100}) == false); // divisible par 100
```

```
std::cout << "Tous les tests sont passés avec succès.\n";
}
```

```
int main() {
    tester_est_bissextile();
    return 0;
}
```

💡 Tu peux maintenant implémenter la fonction `est_bissextile` pour valider ces tests. Si tu veux plus de cas limites ou des variantes, fais-le moi savoir !

## Meilleure approche (3)

J'ai écrit la fonction suivante mais elle ne passe pas les tests, peux-tu me guider vers la solution sans me la donner ?

```
bool est_bissextile(const Date& d) {  
    return d.annee % 4 == 0;  
}
```

Bien sûr, tu es sur la bonne voie, mais ta condition est **trop simpliste** : elle ne tient compte **que de la divisibilité par 4**, ce qui n'est **pas suffisant** selon la règle complète.

Rappelle-toi de la règle :


Une année est bissextile si **elle est divisible par 4, mais pas par 100, sauf si elle est aussi divisible par 400**.

### Quelques pistes pour t'aider à corriger :

- Ton code accepte 1900 comme bissextile, car  $1900 \% 4 == 0$ , mais 1900 **n'est pas** bissextile (divisible par 100 mais pas par 400).
- Ton code n'accepte pas **correctement** les années comme 2000, qui **sont** bissextiles parce qu'elles sont divisibles par 400.

### Questions pour t'aiguiller :

1. As-tu pensé à vérifier si l'année est **divisible par 100** ? Que devrait-on faire dans ce cas ?
2. Quelle est la **priorité** entre "divisible par 4", "divisible par 100" et "divisible par 400" dans la règle ?
3. Comment pourrais-tu traduire cette règle **en if/else**, ou bien en **expressions logiques combinées** ?

Essaie de reformuler la condition complète dans un `if` logique. Si tu veux que je valide ton prochain essai sans te donner la solution,  peux aussi faire ça.



# Meilleure approche (4)



```
bool est_bissextile(const Date& d) {  
    return d.annee % 400 or not d.annee % 100 and  
    d.annee % 4 == 0;  
}
```

Merci pour ton code ! Tu n'es vraiment **pas loin du tout** de la bonne solution — mais il y a **deux erreurs principales** à corriger :

## 🔴 1. Erreur de logique dans l'ordre des priorités

Ton expression :

cpp

📋 Copier 🗑️ Modifier

```
return d.annee % 400 or not d.annee % 100 and d.annee % 4 == 0;
```

se traduit par quelque chose comme :

"Si l'année **n'est pas divisible par 400**, ou **n'est pas divisible par 100** et est **divisible par 4**"

Mais ça, ce n'est **pas** ce que dit la règle officielle.

## ✅ 2. Ce que dit la vraie règle :

Une année est bissextile si :

- elle est divisible par 400, ou
- elle est divisible par 4 et pas divisible par 100

Autrement dit, la **priorité** va aux tests suivants :

scss

📋 Copier 🗑️ Modifier

```
SI (divisible par 400) → bissextile  
SINON SI (divisible par 4 ET pas divisible par 100) → bissextile  
SINON → pas bissextile
```

## 🧠 Indices pour corriger :

- Utilise les **parenthèses** pour forcer l'ordre des opérations dans ton `return` .
- En C++, les opérateurs `and` , `or` , et `not` sont valides, mais ils peuvent prêter à confusion. Tu peux aussi utiliser `&&` , `||` , `!` si tu préfères.

Cela fonctionne-t-il ?

```
bool est_bissextile(const Date& d) {
    return d.annee % 400 == 0 or (not d.annee % 100 ==
0 and d.annee % 4 == 0);
}
```

Tu y es presque ! Mais il y a **encore une petite subtilité piégeuse** dans cette expression, au niveau de la syntaxe logique.

## 📌 Problème : Priorité de `not` mal utilisée

Dans cette partie de ton code :

```
cpp Copier Modifier

not d.annee % 100 == 0
```

tu écris `not d.annee % 100 == 0`, mais cela ne fait **pas** ce que tu crois.

En C++, `not a % b == 0` est interprété comme :

```
cpp Copier Modifier

(not d.annee) % 100 == 0
```

Ce qui est très différent de :

```
cpp Copier Modifier

not (d.annee % 100 == 0)
```

# Meilleure approche (5)



Donc ta condition ne marche pas comme prévu, car `not` est appliqué **avant** le modulo.

## ✅ Correction à faire

Ajoute des **parenthèses explicites** autour de la comparaison, comme ceci :

```
cpp Copier Modifier

return d.annee % 400 == 0 or ((d.annee % 4 == 0) and (d.annee % 100 != 0
```

ou en gardant ton style avec `not` :

```
cpp Copier Modifier

return d.annee % 400 == 0 or (d.annee % 4 == 0 and not (d.annee % 100 ==
```

## ✅ Résultat

Avec cette correction, **ton code respectera parfaitement la règle** et les tests passeront.

# En conclusion ...



- Ce n'est pas l'outil qui compte, c'est dans quel but vous l'utilisez
- Les exercices et laboratoires de PRG1 servent
  - à vous aider à comprendre les bases de la programmation (en C++)
  - à vous entraîner à apprendre à écrire du code par vous-même
  - accessoirement, à vous préparer à réussir les tests et examens de PRG1
- Quelques soient les outils que vous utilisez, assurez-vous qu'ils vous aident à atteindre ces objectifs là.

# Et méfiez vous de cet outil ...



Ton expression :

cpp

Copier Modifier

```
return d.annee % 400 or not d.annee % 100 and d.annee % 4 == 0;
```

se traduit par quelque chose comme :

"Si l'année n'est pas divisible par 400, ou n'est pas divisible par 100 et est divisible par 4"

Mais ça, ce n'est pas ce que dit la règle officielle.

## ✓ 2. Ce que dit la vraie règle :

Une année est bissextile si :

- elle est divisible par 400, ou
- elle est divisible par 4 et pas divisible par 100

Autrement dit, la **priorité** va aux tests suivants :

scss

Copier Modifier

```
SI (divisible par 400) → bissextile  
SINON SI (divisible par 4 ET pas divisible par 100) → bissextile  
SINON → pas bissextile
```



- Attention, ici ChatGPT raconte n'importe quoi.
- Le seul problème de cette expression était la présence du **not**. Sans lui le code est correct
- Il recommande d'ajouter des parenthèses qui ne font que confirmer la priorité par défaut des opérateurs **and** et **or** et ne changent pas l'effet du code
- **not** (d.annee % 100) se traduirait par « est divisible par 100 », pas le contraire
- (**not** d.annee) % 100 se traduirait par « annee non nulle »
- En conclusion ... s'ils sont disponibles, posez plutôt vos questions au prof ou à l'assistant ...