Séquence, sélection et itération

https://github.com/HEIG-VD-Prog-Course/HEIG-VD-ProgIM-Course

Support de cours · Présentation (web) · Présentation (PDF)

V. Guidoux, avec l'aide de GitHub Copilot.

Ce travail est sous licence CC BY-SA 4.0.

Retrouvez plus de détails dans le support de cours

Cette présentation est un résumé du support de cours. Pour plus de détails, consultez le support de cours.

Objectifs (1/3)

- Comprendre les trois piliers fondamentaux de la programmation
- Décrire le concept de variable en programmation
- Écrire des algorithmes en pseudocode
- Créer des diagrammes d'activité avec PlantUML

Objectifs (2/3)

- Comprendre et appliquer la séquence d'instructions
- Décrire la sélection avec les conditions SI/ALORS/SINON
- Utiliser l'itération avec les différents types de boucles
- Traduire des concepts algorithmiques en diagrammes visuels

Objectifs (3/3)

- Analyser et décomposer des problèmes complexes

Les variables

Qu'est-ce qu'une variable ?

- Une **boîte** qui peut contenir une valeur
- Comme une **étiquette** sur un tiroir
- Peut changer de contenu au cours du temps
- Essentielle pour stocker et manipuler des données

Variables : exemple concret

```
name = "Marie"
user_age = 20
height = 1.65
is_student = vrai
```

Analogie : Pensez aux variables comme des tiroirs étiquetés dans un bureau.

Le pseudocode

Qu'est-ce que le pseudocode ?

- Langage **simplifié** pour décrire des algorithmes
- **Français structuré** avec des mots-clés
- Indépendant de tout langage de programmation
- Permet de **réfléchir** avant de coder

Règles de base du pseudocode

```
DÉBUT
    Demander le nom de l'utilisatrice
    Sauvegarder le nom dans la variable name

    Afficher "Bonjour", name
FIN
```

- `DÉBUT` et `FIN` délimitent l'algorithme
- Instructions claires en français
- Utilisation de variables explicites

PlantUML : vos diagrammes visuels

Pourquoi utiliser PlantUML ?

- **Visualise** la logique de vos algorithmes
- **Facilite la compréhension** des flux complexes
- **Standard professionnel** dans l'informatique
- **Éditeur en ligne** disponible gratuitement

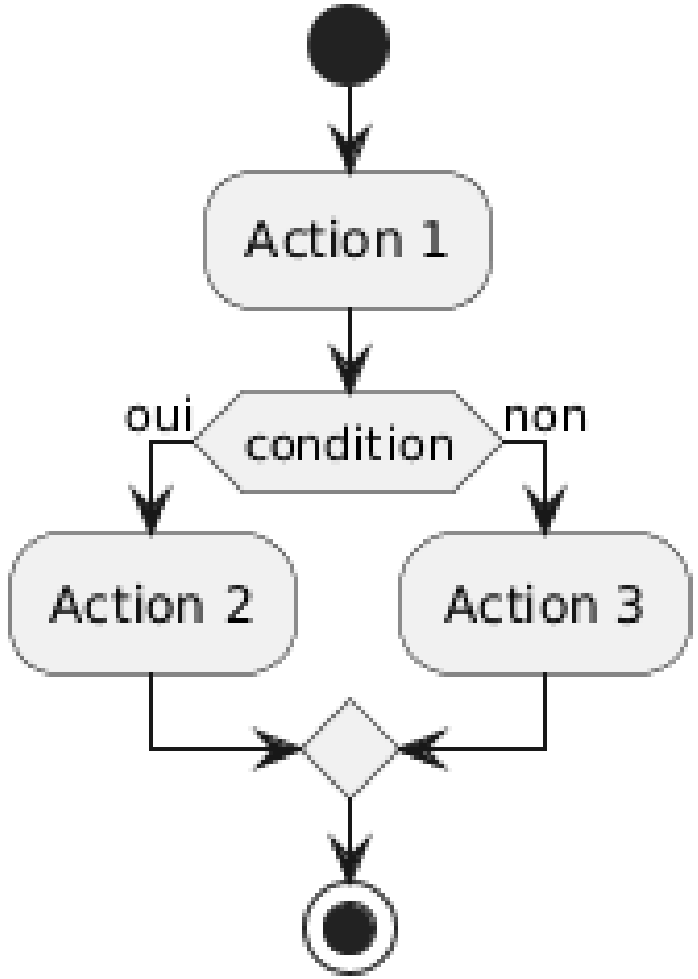
Liens utiles :

- https://www.plantuml.com/plantuml/uml/
- https://plantuml.com/fr/activity-diagram-beta

Éléments de base PlantUML

- `start` / `stop` : début et fin
- `:action;` : une action

```
@startuml
start
:Action 1;
if (condition) then (oui)
    :Action 2;
else (non)
    :Action 3;
endif
stop
@enduml
```



Les trois piliers de la programmation

Tout programme informatique repose sur trois concepts fondamentaux :

1. **Séquence** : exécuter des instructions dans l'ordre
2. **Sélection** : prendre des décisions
3. **Itération** : répéter des actions

La séquence

Qu'est-ce qu'une séquence ?

- Suite d'**instructions exécutées dans l'ordre**
- Chaque instruction se fait **après** la précédente
- Comme une **recette de cuisine** : étape 1, puis 2, puis 3…

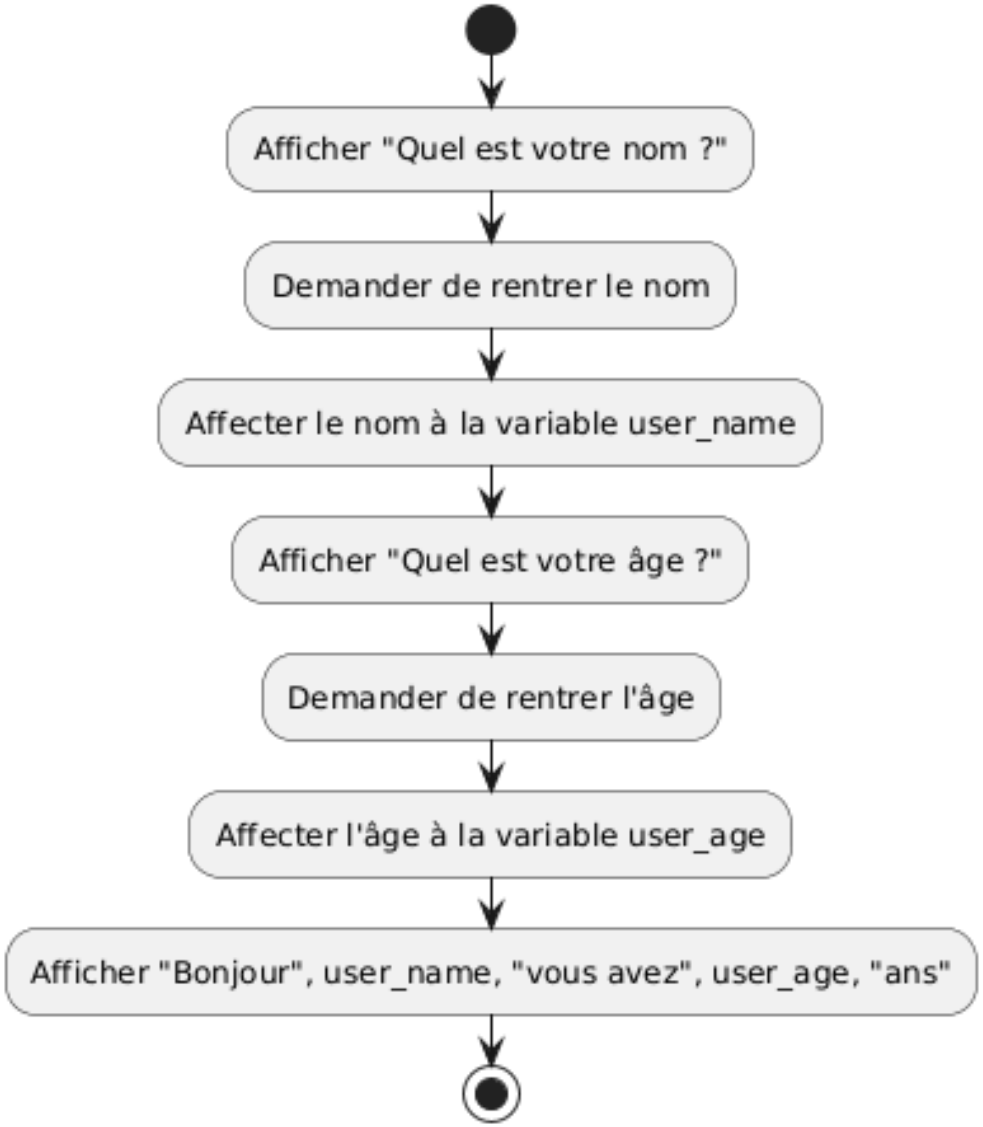
Exemple de séquence

```
DÉBUT
    Afficher "Quel est votre nom ?"
    Demander user_name

    Afficher "Quel est votre âge ?"
    Demander user_age

    Afficher "Bonjour", user_name, "vous avez", user_age, "ans"
FIN
```

Va afficher : "Bonjour Alice, vous avez 30 ans" si l'utilisatrice a entré "Alice" et "30".

Diagramme d'activité - Séquence

```
@startuml sequence
start
:Afficher "Quel est votre nom ?";
:Demander de rentrer le nom;
:Affecter le nom à la variable user_name;
:Afficher "Quel est votre âge ?";
:Demander de rentrer l'âge;
:Affecter l'âge à la variable user_age;
:Afficher "Bonjour", user_name, "vous avez", user_age, "ans";
stop
@enduml
```



Les rectangles montrent la séquence d'actions.

La sélection

Qu'est-ce qu'une sélection ?

- Permet de **prendre des décisions** dans un programme
- **SI** une condition est vraie, **ALORS** faire quelque chose
- **SINON** faire autre chose
- Comme un **carrefour** : aller à gauche ou à droite ?

Structure de la sélection

```
SI (condition) ALORS
    actions si la condition est vraie
SINON
    actions si la condition est fausse
FIN SI
```

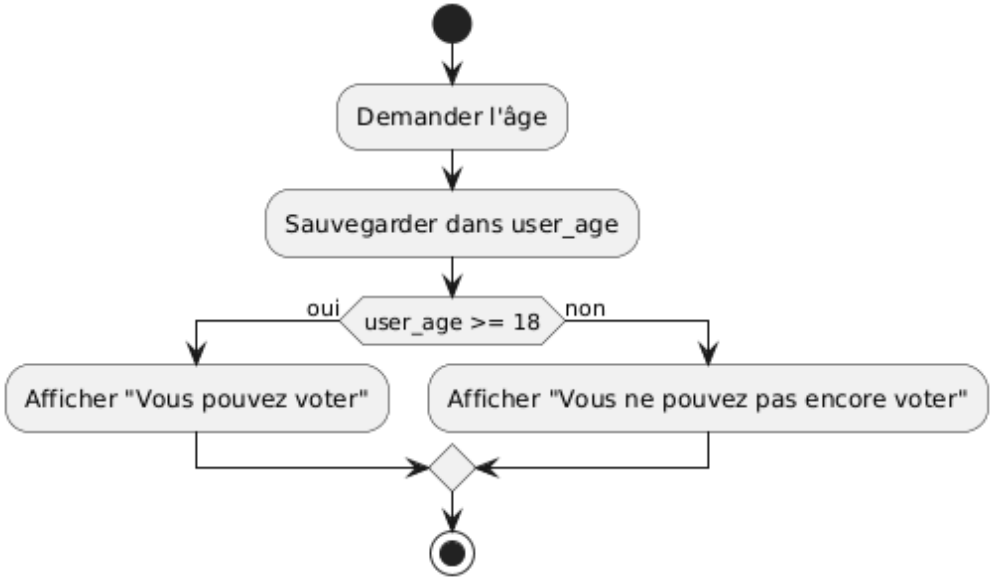
Exemple : Déterminer si une personne peut voter

Exemple concret de sélection

```
DÉBUT
    Demander l'âge de la personne
    Sauvegarder l'âge dans user_age

    SI (user_age >= 18) ALORS
        Afficher "Vous pouvez voter"
    SINON
        Afficher "Vous ne pouvez pas encore voter"
    FIN SI
FIN
```

Diagramme d'activité - Sélection

```
@startuml
start
:Demander l'âge;
:Sauvegarder dans user_age;
if (user_age >= 18) then (oui)
    :Afficher "Vous pouvez voter";
else (non)
    :Afficher "Vous ne pouvez pas encore voter";
endif
stop
@enduml
```



Le losange représente la condition.

L'itération

Qu'est-ce qu'une itération ?

- Permet de **répéter** des instructions plusieurs fois
- Évite de réécrire le même code
- Comme apprendre ses **tables de multiplication** : répéter jusqu'à maîtriser
- Trois types principaux de boucles

Les trois types d'itération

1. **POUR** : quand on sait **combien de fois** répéter
2. **TANT QUE** : répéter **tant qu'une condition** est vraie
3. **RÉPÉTER JUSQU'À** : répéter **jusqu'à ce qu'une condition** soit vraie

Boucle POUR - Exemple

```
DÉBUT
    POUR i DE 1 À 10 FAIRE
        Afficher "Table de 5 :", 5, "×", i, "=", (5 * i)
    FIN POUR
FIN
```

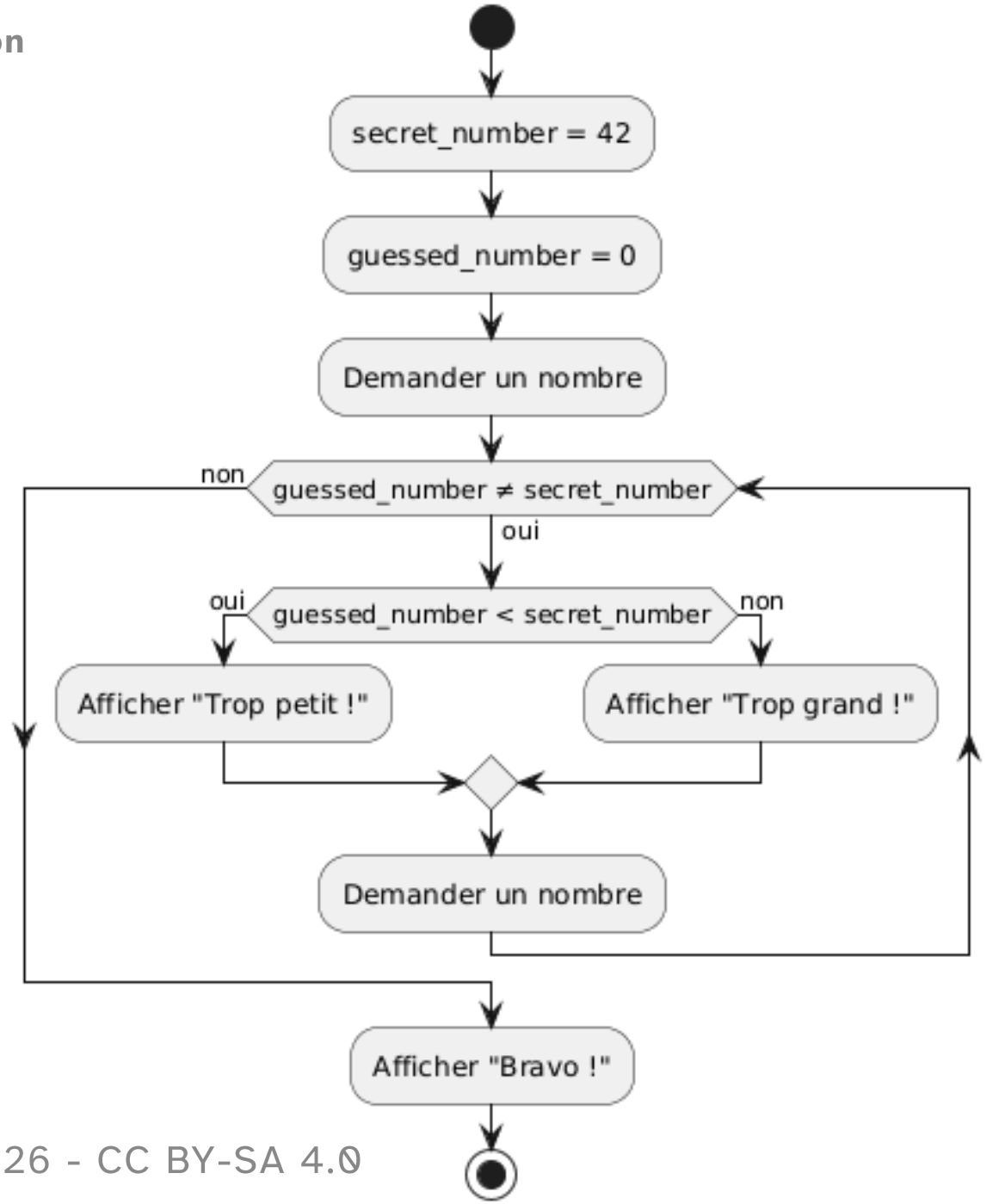
Résultat : Affiche la table de multiplication de 5 de 1 à 10.

Boucle TANT QUE - Exemple

```
DÉBUT
    secret_number = 42
    guessed_number = 0
    Demander un nombre
    TANT QUE (guessed_number ≠ secret_number) FAIRE
        SI (guessed_number < secret_number) ALORS
            Afficher "Trop petit !"
        SINON
            Afficher "Trop grand !"
        FIN SI
        Demander un nombre
    FIN TANT QUE
    Afficher "Bravo !"
FIN
```

Diagramme d'activité - Itération

```
start
:secret_number = 42;
:guessed_number = 0;
:Demander un nombre;
while (guessed_number ≠ secret_number) is (oui)
    if (guessed_number < secret_number) then (oui)
        :Afficher "Trop petit !";
    else (non)
        :Afficher "Trop grand !";
    endif
    :Demander un nombre;
endwhile (non)
:Afficher "Bravo !";
stop
```


secret_number = 42
guessed_number = 0
Demander un nombre
non
guessed_number ≠ secret_number
oui
oui
guessed_number < secret_number
non
Afficher "Trop petit !"
Afficher "Trop grand !"
Demander un nombre
Afficher "Bravo !"

À vous de jouer !

- (Re)lire le support de cours
- Réaliser les exercices
- Expérimenter avec PlantUML en ligne
- Poser des questions si nécessaire

La programmation s'apprend par la pratique ! N'hésitez pas à créer vos propres exercices.

Questions

Est-ce que vous avez des questions ?

Sources

- Illustration objectifs par Aline de Nadai sur Unsplash
- Illustration variables par Javier Allegue Barros sur Unsplash
- Illustration pseudocode par Florian Olivo sur Unsplash
- Illustration exercices par Green Chameleon sur Unsplash
- Illustration récap par Patrick Perkins sur Unsplash
- Illustration à vous de jouer par Nikita Kachanovsky sur Unsplash