UFR MI

Support de cours

INITIATION A L'ALGORITHME

L1 - MIAGE

Equipe Pédagogique Informatique

M. ASSOHOUN Egomli Stanislas

Plan

- CHI – Généralités et structures de contrôle
- CHII – Tableaux
- CHIII – Algorithme procédural
- CHIV- Enregistrements
- CHV- Fichiers

Objectifs:

Dans ce cours, nous allons:

- Décrire ce qu'est un algorithme.
- Décrire les notions des base et les structures de contrôles d'un algorithme
- Appliquer ces notion pour la résolution d'un problème
- Implémenter les algorithme dans le langage Python

INTRODUCTION

L'écriture d'un programme informatique est une tâche exigeante dans sa rigueur. Les ordinateurs exigent perfection et précision dans les instructions qui leur sont données car ils ne font que ce qu'on leur demande de faire. Il ne viendrait à l'idée de personne de rédiger une dissertation sans au préalable construire un plan. Des idées mêmes bonnes ne suffissent pas à faire une bonne rédaction, une structure est nécessaire à leur cohésion. Il en est de même en informatique, faute de méthodes nombres de programmes ne fonctionnent jamais ou le font très mal. Apprendre à programmer consistes plus à acquérir une démarche d'esprit, une méthodologie, qu'à connaître les tours et détours d'un langage de programmation. Connaître un langage de programmation n'est pas savoir programmer. Programmer, c'est définir précisément le problème à résoudre, décrire peu à peu une solution et après seulement l'exprimer dans un langage de programmation. Dans cette démarche l'activité la plus créative n'est pas le codage mais l'analyse du problème. L'algorithmique doit être perçu comme un outil favorisant la réflexion, l'analyse d'un problème tout en gommant les difficultés inhérentes aux langages de programmation.

CH I – Généralités et structures de contrôle

CH I - Généralités et structures de contrôle

I- Généralités

I-1- Historique : Le mot « algorithme » provient de la forme latine (Algorismus) du nom du mathématicien arabe ALKHAREZMI

I-2- Définitions

- **Algorithme** : Un algorithme est une suite ordonnées et finie d'instructions élémentaires permettant de résoudre un problème. Il reçoit en entrée des données pour produire des résultats.
- **Algorithme :** Un algorithme est un pseudo-langage utilisé comme interface entre les langages d'humain et les langages de machine.

- **Algorithmique**: L'algorithmique est la science qui étudie les algorithmes.

CH I - Généralités et structures de contrôle

I- Généralités

I-3- Notions de base

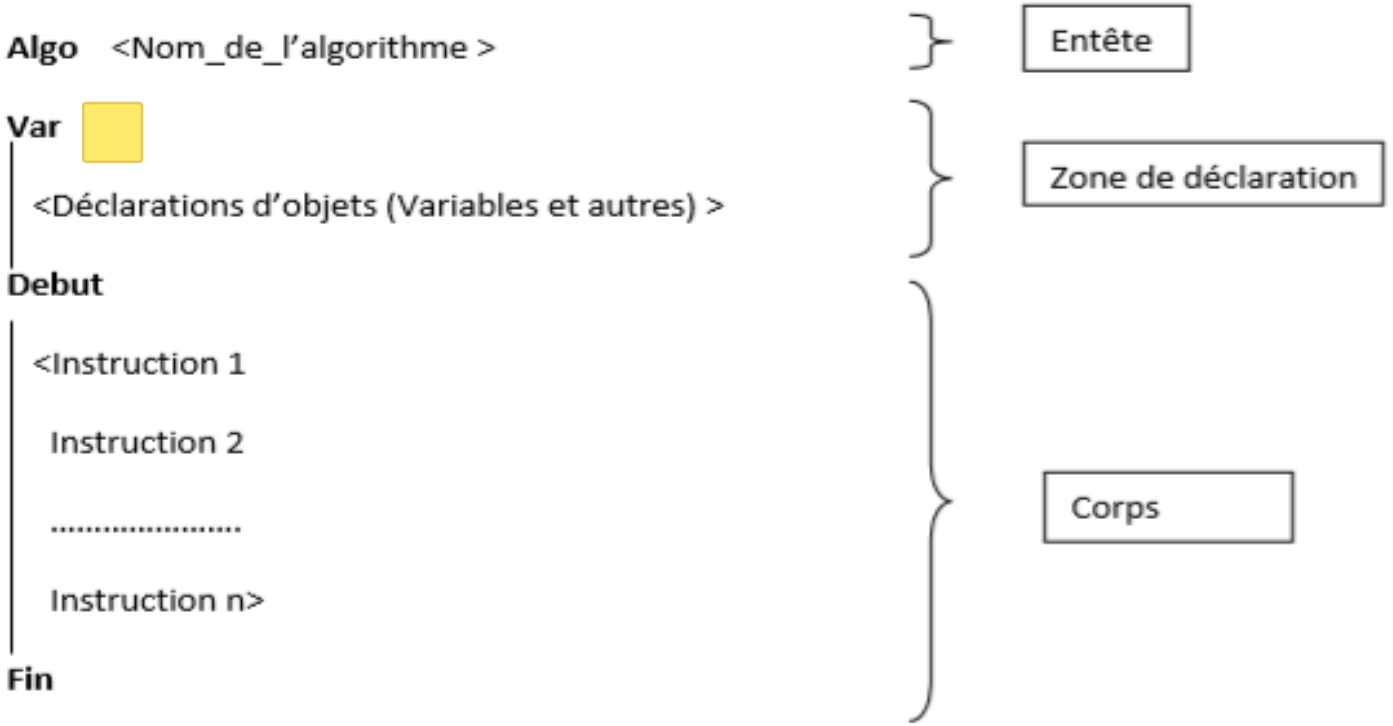
- **théorie de la calculabilité** (Turing-Church): les problèmes ayant une solution algorithmique sont ceux résolubles par une machine de Turing
- **Troisième loi de Greer** : un programme informatique ne fait jamais ce que vous voudriez qu'il fasse …, il fait seulement ce que vous lui dites de faire.

- **Résumer :**
  - ✓ **Algorithme :** Un algorithme est un procédé calculatoire sur un problème donné en nombre fini d'instructions pour obtenir des résultats..
  - ✓ **Programmation**: La programmation est l'activité de traduire dans un langage de programmation un algorithme..

CH I - Généralités et structures de contrôle

I- Généralités

I-4- Structure générale d'un algorithme séquentiel



Description :

Nom_de_l'algorithme : C'est un identifiant symbolique que le programmeur donne à chacun de ses algorithmes.

Ex : **Algo** somme2nbre

pour un algorithme calculant la somme de 2 nombres

CH I - Généralités et structures de contrôle

I- generalités

I-5- Les variables

- Une variable est une structure de données utilisée pour stocker des données en mémoire.

- La déclaration des variables correspond à la réservation en mémoire d'un espace pour toutes les données utilisées dans notre programme.

- **Syntaxe**:

Nom_de_la_variable : Type_de_la_variable

CH I - Généralités et structures de contrôle

I- généralités

I-5- Les variables

Nom_de_la_variable est un identifiant symbolique utilisé par le programmeur pour désigner chaque variable dans son programme.

Ex : pour une variable exprimant par exemple :

- **Le résultat → Res** : entier
- **L'opérant 1 → op1** : entier
- **L'opérant 1 → op1** : entier

Variable	espace mémoire
op1	
op2	
res	

NB: la réservation de variable correspond à une réservation d'un espace mémoire vide

CH I - Généralités et structures de contrôle

I- Généralités

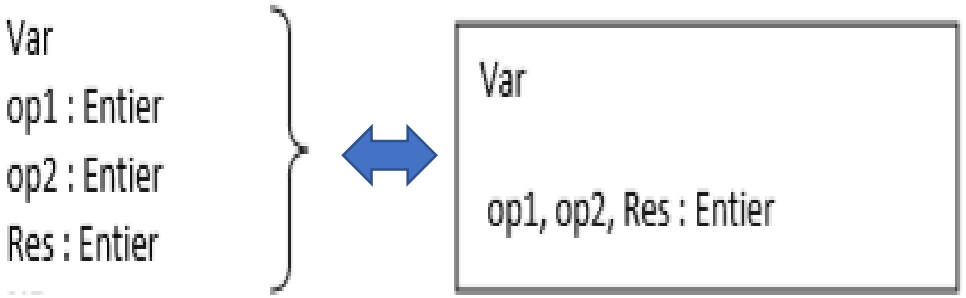
I-5- Les variables

Type_de_la_variable, c'est le type de données d'une variable.

Les types couramment utilisés sont :

- **Types simples :** Entier, Réel, Caractère, Booléen
- **Types complexes :** Chaîne (de caractères), Tableau, Enregistrement,……

Exemple de déclaration de variables :



NB:

lorsque plusieurs variables sont du même type, on peut faire la déclaration sur une même ligne.

CH I - Généralités et structures de contrôle

I- Généralités

I-5- Les Constantes

Certaines informations manipulées par un programme ne changent jamais. C'est par exemple la cas de la valeur de π, du nombre maximum d'étudiants dans une promotion, etc. Ces données ne sont donc pas variables mais constantes. Plutôt que de mettre explicitement leur valeur dans le texte du programme (constantes littérales), il est préférable de leur donner un nom symbolique (et significatif). On parle alors de constantes (symboliques).

Syntaxe: CONST nom_const = valeur

Exemple:
CONST PI = 3.1415 ou CONST INTITULÉ = "Algorithmique et programmation"

Remarque : les constantes sont mises en place avant la section déclaration des variables.

CH I - Généralités et structures de contrôle

I- Généralités

I-6- Les instructions

- **L'affectation :** Elle est utilisée pour renseigner un espace mémoire préalablement réservé. Il s'agit d'affecter un contenu, une valeur à une variable sans passer par le clavier.

Ex : op1←4 (opérante 1 reçoit 4)
op2←5 (opérante2 reçoit 5)
Res←op1 + op2 (Résultat reçoit « opérante1 + opérante2 »)

Variable	espace mémoire
op1	4
op2	5
res	9

- **Opérateurs arithmétiques** : (+,-,*,/, Div (division entière, ex : 7 Div 2 = 3), Mod (ex : 7 Mod 2 = 1)
- **Opérateurs relationnels** : (=, <, <=, >, >=, <>)

CH I - Généralités et structures de contrôle

I- Généralités

I-6- Les instructions

- **Opérateurs logiques :** Un operateur logique est un operateur qui agit sur les variables de type booléen et le résultat est celui de la table de vérité

ET

P	Q	P ET Q
V	V	V
F	V	F
V	F	F
F	F	F

OU(OU inclusif)

P	Q	P OU Q
V	V	V
F	V	V
V	F	V
F	F	F

XOU (OU exclusif)

P	Q	P OU Q
V	V	F
F	V	V
V	F	V
F	F	F

NON

P	NON P
V	F
F	V

CH I - Généralités et structures de contrôle

I- Généralitésse

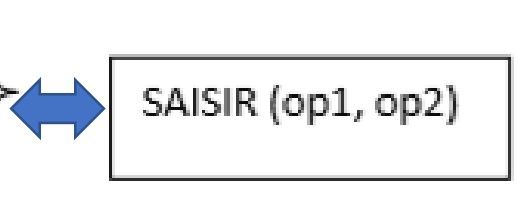
I-6- Les instructions

- **Entrées et Les Sorties :** Pour entrer une valeur au clavier, on utilise l'instruction SAISIR.

Syntaxe : SAISIR(Nom_de_la_variable)

Exemple
SAISIR (op1)

SAISIR (op2)



- **Sorties** Pour afficher quelque chose à l'écran on utilise l'instruction : AFFICHER

Syntaxe : AFFICHER(" message") **AFFICHER("Entrer le premier nombre")**

Syntaxe : AFFICHER(Nom_de_la_variable) **AFICHER(res)**

CH I - Généralités et structures de contrôle

I- Généralités

I-6- Les instructions

❑ Sorties (suite)

- ✓ AFFICHER (“message à afficher ”)
- ✓ AFFICHER(Nom_de_la_variable)
- ✓ AFFICHER (“message à afficher ”, Nom_de_la_variable)
- ✓ Ex: AFFICHER(”Le resultat est”,res) Le resultat est 9

Exemple :
AFFICHER (“Entrez deux nombres”)
AFFICHER (Res)
AFFICHER (”La somme est : ”, Res)

CH I - Généralités et structures de contrôle

I- Généralités

I-7- Démarche de résolution d'un problème

Problème : Ecrire un algorithme qui calcule et affiche la somme de 2 entiers saisis au clavier par l'utilisateur

- ❑ **Etape 1** : L'Analyse « mathématique »

Faire une représentation mathématique de la solution si possible

Res = op1 + op2

- ❑ **Etape 2** : Les objectifs

Décrire le ou les besoins à atteindre

- ✓ Saisir 2 entiers au clavier
- ✓ Calculer la somme des 2 entiers
- ✓ Afficher le résultat

CH I - Généralités et structures de contrôle

I- Généralités

I-7- Démarche de résolution d'un problème (suite)

❑ **Etape3 :** Les besoins

Dresser la liste des besoins (données/variables/ objets que le programmes va utiliser)

Variables	Type	Règle de calcul
Op1	Entier	
Op2	Entier	
Res	Entier	Res←op1+op2

CH I - Généralités et structures de contrôle

I- Généralités

I-7- Démarche de résolution d'un problème (suite)

- **Etape4** : Ecrire l'algorithme

```
Algo   Somme
 Var
   op1, op2, Res : Entier
Début
   AFFICHER ("Entrez 2 entiers SVP !")
   SAISIR (op1, op2)
   Res ← op1 + op2
   AFFICHER ("La somme est : ", Res)
Fin
```

CH I - Généralités et structures de contrôle

I- Généralités

I-7- Démarche de résolution d'un problème (suite)

❑ **Etape 5 :** Faire la trace

Cette action consiste à exécuter le programme à la main. Il s'agit de simuler et prévoir à la main le comportement de la machine.

N°	Op1	Op2	Res	Ecran
1	//	//	//	Entrez 2 entiers SVP !
2	4	5		4 (OK : Touche Entrer du clavier) 5 (OK)
3	4	5	9	
4	4	5	9	La somme est : 9

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-1- Présentation

En algorithme structurée, le transfert de contrôle s'exprime par :

1. enchaînement séquentiel (une instruction puis la suivante);
2. traitements conditionnels;
3. traitements répétitifs (itératifs);
4. appel d'un sous-programme (un autre programme qui réalise un traitement particulier).

Pour chacune des structures de contrôle présentées ci-après, sont données la syntaxe de la structure dans notre langage algorithmique, les règles à respecter (en particulier pour le typage), la sémantique (c'est-à-dire la manière dont elle doit être comprise et donc interprétée), des exemples ou exercices à but illustratifs.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-1- Enchaînement séquentiel

Les instructions sont exécutées dans l'ordre où elles apparaissent.

```
1       opération_1
2       ...
3       opération_n
```

Exemple

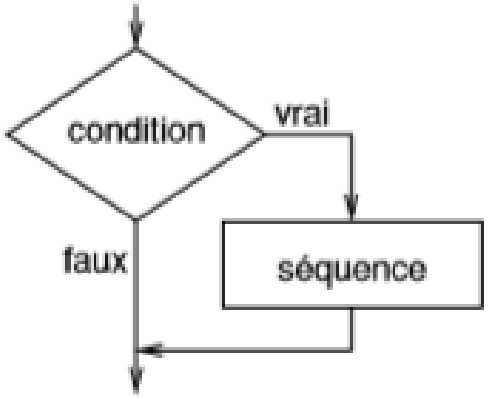
```
1       tmp <- a        -- première instruction
2       a <- b          -- deuxième instruction
3       b <- tmp        -- troisième instruction
```

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-2- Instructions conditionnelles

❑ Conditionnelle Si ... Alors ... FinSi



Si condition Alors

séquence -- une séquence d'instructions

FinSi

Règle: La condition est nécessairement une expression booléenne.

Évaluation:

– la condition est évaluée;

– si la condition est vraie, la séquence est exécutée puis le contrôle passe à l'instruction qui suit le FinSi;

En d'autres termes, la séquence est exécutée si et seulement si la condition est VRAI.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-2- Instructions conditionnelles

❑ Conditionnelle Si ... Alors ... Sinon ... FinSi

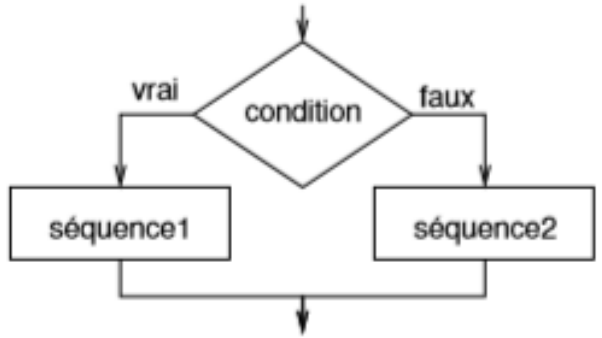
Si condition Alors

séquence1 -- séquence exécutée ssi condition est VRAI

Sinon { Non condition }

séquence2 -- séquence exécutée ssi condition est FAUX

FinSi



Évaluation:

Si la condition est vraie, c'est séquence1 qui est exécutée, sinon c'est séquence2. Dans les deux cas, après l'exécution de la séquence, l'instruction suivante à exécuter est celle qui suit le FinSi.

CH II- Généralités et structures de contrôle

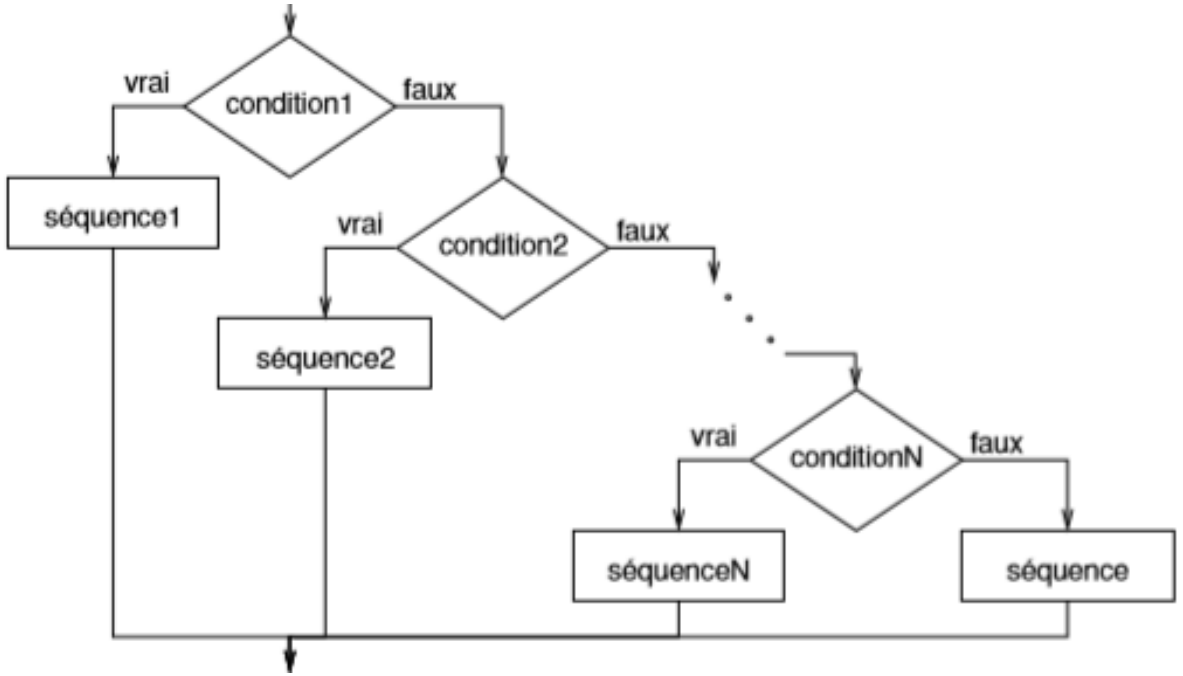
II- Structures de contrôle

II-2- Instructions conditionnelles

❑ **La clause SinonSi**

La conditionnelle Si ... Alors ... Sinon ... FinSi peut être complétée avec des clauses SinonSi suivant le schéma suivant :

Si condition1 Alors
séquence1
SinonSi condition2 Alors
séquence2
...
SinonSi conditionN Alors
séquenceN
Sinon { Expliciter la condition ! }
séquence 10
FinSi



CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-2- Instructions conditionnelles

❑ **La clause SinonSi (suite)**

Évaluation :
Les conditions sont évaluées dans l'ordre d'apparition. Dès qu'une condition est vraie, la séquence associée est exécutée. L'instruction suivante à exécuter sera alors celle qui suit le FinSi.
Si aucune condition n'est vérifiée, alors la séquence associée au Sinon, si elle existe, est exécutée.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-2- Instructions conditionnelles

❑ **Conditionnelle Selon**

```
Selon expression Faire
        choix1 : séquence1
        choix2 : séquence2
        ...
        choixN : séquenceN
        [Sinon séquence 11 ]
FinSelon
```

Règle :

– expression est nécessairement une expression de type scalaire.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-2- Instructions conditionnelles

❑ Conditionnelle Selon (suite)

Règle :

– expression est nécessairement une expression de type scalaire.

– $choix_i$ est une liste de choix séparés par des virgules. Chaque choix est soit une constante, soit un intervalle (10..20, par exemple).

L'instruction Selon peut donc être considérée comme un cas particulier de la conditionnelle Si ... SinonSi ... FinSi.

Évaluation :

L'expression est évaluée, puis sa valeur est successivement comparée à chacun des ensembles $choix_i$. Dès qu'il y a correspondance, les comparaisons sont arrêtées et la séquence associée est exécutée. Les différents choix sont donc exclusifs. Si aucun choix ne correspondant, alors la séquence associée au Sinon, si elle existe, est exécutée.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

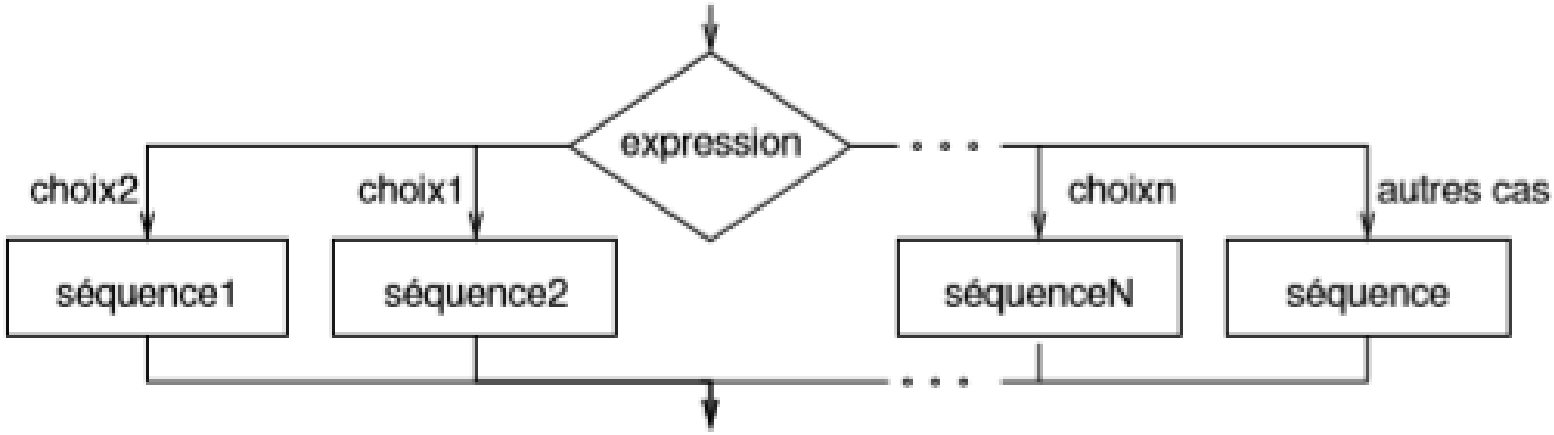
II-2- Instructions conditionnelles

❑ **Conditionnelle Selon (suite)**

Remarque :

La clause Sinon est optionnelle... mais il faut être sûr de traiter tous les cas (toutes les valeurs possibles de l'expression).

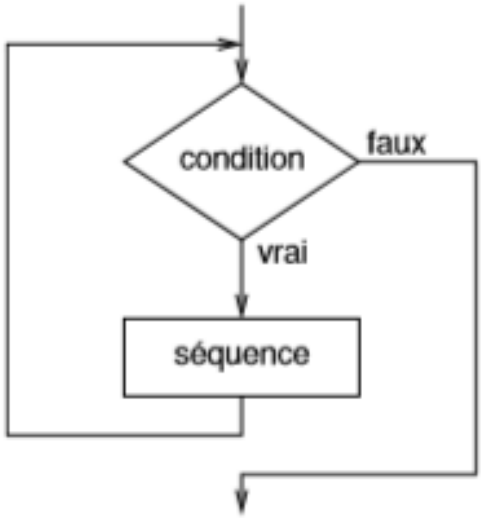
Organigramme :



CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions



❑ **Répétition TantQue**

TantQue condition Faire
    séquence
FinTQ

Règles :

– La condition doit être une expression booléenne.

– Pour que la boucle se termine, il est nécessaire que la séquence modifie la condition.

Évaluation:

La condition est évaluée. Si elle vaut FAUX alors la boucle se termine et l'exécution se poursuit avec l'instruction qui suit FinTQ. Si elle vaut VRAI alors la séquence d'instructions est exécutée et la condition est de nouveau évaluée.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

- **Répétition TantQue (suite)**

Remarque : Comme le test de la condition est fait en premier, la séquence peut ne pas être exécutée. Il suffit que la condition soit fausse dès le début.

Problème de la terminaison.

Nous avons vu qu'une propriété importante d'un algorithme/programme est qu'il doit se terminer. Jusqu'à présent la terminaison était assurée car avec seulement la séquence et les conditionnelles, l'exécution du programme se fait toujours vers l'avant. On doit donc nécessairement atteindre la fin. Avec les répétitions, on peut revenir en arrière dans le programme. Il est alors important de s'assurer que l'on finira toujours par sortir des répétitions d'un programme. Sinon, leprogramme risque de s'exécuter indéfiniment. On parle alors de boucle sans fin. Pour garantir qu'une boucle (une répétition) se termine, on peut mettre en évidence un variant. C'est une expression entière qui doit toujours être positive et décroître strictement à chaque passage dans la boucle.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

- **Répétition TantQue (suite)**

Problème de la terminaison. (suite)

Elle aide à prouver la validité d'une boucle et d'un programme. L'invariant est cependant généralement difficile à trouver et la preuve difficile à faire.

- **Théorème :**

Tout algorithme (calculable) peut être exprimé à l'aide de l'affectation et des trois structures Si Alors FinSi, TantQue et enchaînement séquentiel. ...

Mais ce n'est pas forcément pratique, aussi des structures supplémentaires ont été introduites.

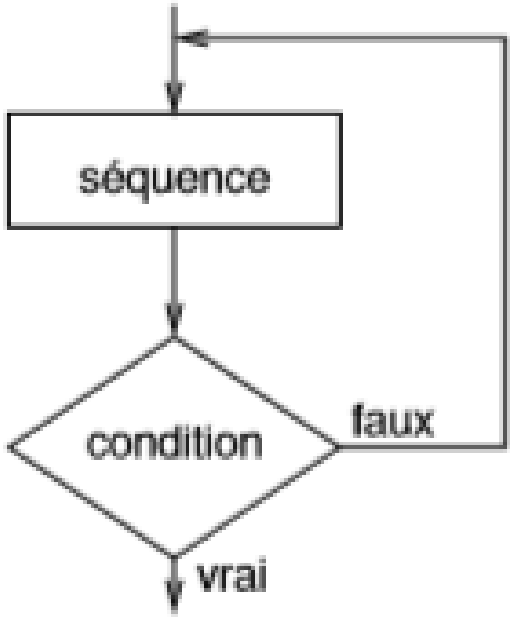
CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

- **Répétition Répéter...JusquÀ**

Répéter
    séquence
JusquÀ condition



Remarques:

– la condition n'est évaluée qu'après l'exécution de la séquence;
– la séquence est exécutée au moins une fois;
– la condition doit être modifiée par la séquence;

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

- **Répétition Pour**

```
Pour var <- val_min [ Décrémenter ] JusquÀ var = val_max Faire
        sequence
FinPour
-- Une variante
Pour var Dans val_min..val_max [ Renversé ] Faire
        sequence
FinPour
```

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

❑ Répétition Pour (suite)

Règle:

– La variable var est une variable d'un type scalaire. Elle est dite variable de contrôle.
– Les expressions val_min et val_max sont d'un type compatible avec celui de var.
– La séquence d'instructions ne doit pas modifier la valeur de la variable var.

Évaluation : Les expressions val_min et val_max sont évaluées. La variable var prend alors successivement chacune des valeurs de l'intervalle [val_min..val_max] dans l'ordre indiqué et pour chaque valeur, la séquence est exécutée.

Remarques:

– Cette structure est utilisée lorsqu'on connaît à l'avance le nombre d'itérations à faire.
– Les expressions val_min et val_max ne sont évaluées qu'une seule fois.
– La séquence peut ne pas être exécutée (intervalle vide : val_min > val_max).
– La séquence termine nécessairement (le variant est val_max - var + 1).

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

❑ Répétition Pour (suite)

Attention:

Il est interdit de modifier la valeur de la variable de contrôle var dans la boucle.

Remarque :

Il n'y a pas d'équivalent du Pour dans les organigrammes. On peut utiliser la traduction du Pour sous forme de TantQue ou de Répéter.

❑ Quelle répétition choisir?

La première question à se poser est : « Est-ce que je connais a priori le nombre d'itérations à effectuer? ». Dans l'affirmative, on choisit la boucle Pour. Dans la négative, on peut généralement employer soit un TantQue, soit un Répéter.

En général, utiliser un Répéter implique de dupliquer une condition et utiliser un TantQue implique de dupliquer une instruction.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

- **Quelle répétition choisir? (suite)**

Dans le cas, où on choisit d'utiliser un Répéter, il faut faire attention que les instructions qu'il contrôle seront exécutées au moins une fois.

S'il existe des cas où la séquence d'instructions ne doit pas être exécutée, alors il faut utiliser un TantQue (ou protéger le Répéter par un Si).

Une heuristique pour choisir entre TantQue et Répéter est de se demander combien de fois on fait l'itération.

Si c'est au moins une fois alors on peut utiliser un Répéter sinon on préfèrera un TantQue.

CH II- Généralités et structures de contrôle

II- Structures de contrôle

II-3- Instructions de répétitions

❑ **Quelle répétition choisir? (suite)**

Remarque: Pour aider au choix, il est parfois judicieux de se poser les questions suivantes :

– Qu'est ce qui est répété (quelle est la séquence)?

– Quand est-ce qu'on arrête (ou continue)?

```
Si nombre d'itérations connu Alors
        Résultat <- Pour
Sinon
        Si itération exécutée au moins une fois Alors
                Résultat <- Répéter
        Sinon
                Résultat <- TantQue
        FinSi
FinSi
```

Merci de votre attention