

COURS ALGORITHMIQUE ET PROGRAMMATION INFORMATIQUE

DUT INFORMATIQUE
S1

[Marie-Agnès peraldi-frati](#)
[Maître de conférences en informatique](#)
[UNS/IUT de Nice côte d'azur](#)

MAP@UNICE.FR

MAP - UNS

1

RÉFÉRENCES

- Algorithmes D.E Knuth CSLI Publications 2011
- Introduction à la science informatique G. Dowek Ed RPA 2010
- Éléments pour une histoire de l'informatique, D.E Knuth CSLI Publications 2011
- Cours et exercices corrigés d'algorithmique- J. Julliard Ed Vuibert Fev 2010
- Algorithmique méthodes et modèles , P Lignelet Ed Masson 1988
- Cours algorithme Cécile Balkanski, Nelly Bensimon, Gérard Ligozat IUT Orsay

MAP - UNS

2

OBJECTIF DU COURS API

- **Notions de base** en algorithmique
- **Types de données** et lien avec la machine
- Notion de **sous-programmes** et lien avec la compilation
- **Qualité**
 - nommage des variables, assertions, documentation ...,
 - pré et post conditions
- **Structures algorithmiques** fondamentales: .
- Implantation des algorithmes dans un langage de programmation.
- **Introduction au test unitaire**, boîte noire,
- **Algorithmes fondamentaux** de recherche recherche d'un élément, parcours, tri, ...
- Avoir une première notion des **performances des algorithmes** utilisés

MAP - UNS

3

NOTION DE BASE EN ALGORITHMIQUE

4

MAP - UNS

CONCEPTS IMPORTANTS EN INFORMATIQUE

- **Algorithme** : mot dérivé du nom du mathématicien al_Khwarizmi qui a vécu au 9ème siècle, était membre d'une académie des sciences à Bagdad .
- Un algorithme prend des **données en entrée**, **exprime un traitement** particulier et fournit des **données en sortie**.
- **Programme** : série d'instructions pouvant s'exécuter en séquence, ou en parallèle (parallélisme matériel) qui réalise (**implémente**) un algorithme

MAP - UNS

5

POURQUOI UN COURS D' "ALGO" ?

- **Pour obtenir** de la «machine» qu'elle effectue un travail à notre place
- **Problème**: expliquer à la «machine» comment elle doit s'y prendre
- **Besoins** :
 - savoir **explicit**er son raisonnement
 - savoir **formal**iser son raisonnement
 - concevoir (et écrire) des **algorithmes**:
 - séquence d'instructions qui décrit comment résoudre un problème particulier

MAP - UNS

6

ALGORITHME

- **Savoir expliquer** comment faire un travail sans la moindre ambiguïté
- **langage simple** : des instructions (pas élémentaires)
- suite finie d'actions à entreprendre en respectant une chronologie imposée
- L'écriture algorithmique : un travail de programmation à visée universelle
 - un algorithme ne **dépend pas du langage** dans lequel il est implanté,
 - ni de la **machine** qui exécutera le programme correspondant.

MAP - UNS

7

EXEMPLE D'ALGORITHMES

- **Recette de cuisine**
- **Notice de montage** de meuble en kit



Recette de la pâte à crêpes

Il te faut :
250 g farine 50 cl 3 oeufs 1 pincée de sel

1 Verse la farine et le sel dans le saladier.

2 Casse les oeufs, mélange avec la cuillère en bois et ajoute progressivement le lait sans cesser de tourner.

3 Laisse reposer la pâte pendant 1 heure.

4 Fais chauffer la poêle. Verse-y une louche de pâte. Répète-le bien bougeant la poêle. Fais cuire une crêpe fine.

Petit monde

- Mathématiques : **problème $3n+1$** : élémentaire mais redoutable
- si n est pair, on le divise par 2 ;
- si n est impair, on le multiplie par 3 et on ajoute 1.
- Est-il vrai que l'on finira tôt ou tard par tomber sur 1 ?

MAP - UNS

8

LES PROBLÈMES FONDAMENTAUX EN ALGORITHMIQUE

- **Complexité**
 - En combien de temps un algorithme va -t-il atteindre le résultat escompté?
 - De quel espace a-t-il besoin?
- **Calculabilité:**
 - Existe-t-il des tâches pour lesquelles il n'existe aucun algorithme ?
 - Étant donnée une tâche, peut-on dire s'il existe un algorithme qui la résolve ?
- **Correction**
 - Peut-on être sûr qu'un algorithme réponde au problème pour lequel il a été conçu ?

MAP - UNS

9

EXEMPLE DE LANGAGE ALGORITHMIQUE

Algorithme ElèveAuCarré

{Cet algorithme calcule le carré du nombre que lui fournit l'utilisateur}

```
variables unNombre, sonCarré: entiers      {déclarations: réservation
                                           d'espace-mémoire}
début                                     {préparation du traitement}
    afficher("Quel nombre voulez-vous élever au carré?")
    saisir(unNombre)
                                           {traitement : calcul du carré}
    sonCarré ← unNombre × unNombre
                                           {présentation du résultat}
    afficher("Le carré de ", unNombre)
    afficher("c'est ", sonCarré)
fin
```

MAP - UNS

10

ETAPES D'UN ALGORITHME

- **Préparation du traitement**
 - données nécessaires à la résolution du problème
- **Traitement**
 - résolution pas à pas,
 - après décomposition en sous-problèmes si nécessaire
- **Edition des résultats**
 - impression à l'écran,
 - dans un fichier, etc.

MAP - UNS

11

LANGAGE ALGORITHMIQUE

Algorithme NomAlgorithme

{ ceci est un commentaire }

Début

... Actions

Fin

Algorithme Bonjour

{il dit juste bonjour mais ... en anglais !}

Début

afficher('Hello world !!!')

ALA ligne

Fin

- Il faut avoir une **écriture rigoureuse**
- Il faut avoir une écriture soignée : respecter l'**indentation**
- Il est nécessaire de **commenter** les algorithmes
- **Il existe plusieurs solutions algorithmiques à un problème posé**
 - ☐ Il faut rechercher l'**efficacité** de ce que l'on écrit

MAP - UNS

12

DÉCLARATION DES DONNÉES

- **Variable** *<nom de donnée>*: **type**
- Instruction permettant de réserver de l'espace mémoire pour stocker des données
- Dépendant du type des données : entiers, réels, caractères, etc.)
- Exemples :
 - **Variables** val, unNombre: **entiers**
nom, prénom : **chaînes de caractères**

MAP - UNS

13

DÉCLARATION DES DONNÉES

- **Constante** *<nom de donnée>*: **type** ← **valeur ou expression**
- Instruction permettant de réserver de l'espace mémoire pour stocker une constante dont la valeur ne varie pas.
- Exemples :
 - **Constante** MAX : entier ← 10
DEUXFOISMAX : entier ← MAX x 2

MAP - UNS

14

LECTURE ÉCRITURE DE DONNÉES

- **Saisir**<nom de donnée, ...>
- **Afficher**<nom de donnée, ...>
- **Fonction** : Instructions permettant
 - de **placer en mémoire** les informations fournies par l'utilisateur.
 - **De visualiser** des données placées en mémoire
- **Exemples:**
 - Saisir**(unNombre)
 - Afficher** (« le nom est », nom, »et le prénom est », prénom)
 - Saisir**(val)

MAP - UNS

15

PHASE D'ANALYSE

- Consiste à extraire de l'énoncé du problème des éléments de modélisation
- Technique : Distinguer en soulignant de différentes couleurs quelles sont
 - Quel est le but du programme (**traitement à réaliser**)
 - **Données en entrée** du problème :
 - Où vont se situer les **résultats en sortie**

MAP - UNS

16

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

- On souhaite calculer et afficher , à partir d'un prix hors taxe saisi, la TVA ainsi que le prix TTC
- Le montant TTC dépend de :
 - Du prix HT
 - Du taux de TVA de 20,6

MAP - UNS

17

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

- On souhaite calculer et afficher , à partir d'un prix hors taxe saisi, la TVA ainsi que le prix TTC
- Le montant TTC dépend de :
 - Du prix HT
 - Du taux de TVA de 20,6

Traitement à réaliser

MAP - UNS

18

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

- On souhaite calculer et afficher , à partir d'un prix hors taxe saisi, la TVA ainsi que le prix TTC
- Le montant TTC dépend de :
 - Du prix HT
 - Du taux de TVA de 20,6

Données en entrée

MAP - UNS

19

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

- On souhaite calculer et afficher , à partir d'un prix hors taxe saisi, la TVA ainsi que le prix TTC
- Le montant TTC dépend de :
 - Du prix HT
 - Du taux de TVA de 20,6

Données en sortie

MAP - UNS

20

ALGORITHME TVA

Algorithme CalculTVA

{Saisit un prix HT et affiche le prix TTC correspondant}

Constantes (TVA : réel) $\leftarrow 20.6$
(Titre : chaîne) \leftarrow "Résultat"

Variables prixHT : réel

Variable prixTTC, montantTVA : réels {déclarations}

Début {préparation du traitement}

afficher("Donnez-moi le prix hors taxe :")

saisir(prixHT)

$\text{prixTTC} \leftarrow \text{prixHT} * (1 + \text{TVA}/100)$ {calcul du prix TTC}

$\text{montantTVA} \leftarrow \text{prixTTC} - \text{prixHT}$

→ Code peu efficace

afficher(Titre) {présentation du résultat}

afficher(prixHT, «euros H.T. + TVA »,TVA, « devient » ,prixTTC,
«eurosT.T.C.»)

Fin

MAP - UNS

21

INSTRUCTIONS SÉQUENTIELLES RÉSULTAT D'UN ALGORITHME

Constante (SEUIL : réel) $\leftarrow 13.25$

Variables valA, valB: réels
compteur : entier
mot , tom : chaînes

$\text{valA} \leftarrow 0.56$

$\text{valB} \leftarrow \text{valA}$

$\text{valA} \leftarrow \text{valA} * (10.5 + \text{SEUIL})$

$\text{compteur} \leftarrow 1$

$\text{compteur} \leftarrow \text{compteur} + 10$

$\text{mot} \leftarrow$ " Bonjour "

$\text{tom} \leftarrow$ "Au revoir ! "

Quelles sont les différentes valeurs des variables ?

MAP - UNS

22

SIMULATION D'UN ALGORITHME

Algorithme CaDoitEchanger?

{Cet algorithme}

Variables valA, valB: **réels** {déclarations}

Début {préparation du traitement}

Afficher ("Donnez-moi deux valeurs :")

Saisir (valA, valB)

Afficher ("Vous m'avez donné ", valA, " et ", valB)

 {traitement mystère}

 valA ← valB

 valB ← valA {présentation du résultat}

Afficher ("Maintenant , mes données sont : ", valA, " et ", valB)

Fin

Que fait cet algorithme ? Pas ce qui est prévu !

MAP - UNS

23

CE QU'IL MANQUE

- Déclarer une variable supplémentaire

Variables valA, valB, valTemp: **entiers**

- Utiliser cette variable pour stocker provisoirement une des valeurs

Saisir(valA, valB)

valTemp ← valA

valA ← valB

valB ← valTemp

MAP - UNS

24

STRUCTURE ALTERNATIVE

« SI ... ALORS ... SINON ... FSI » (1)

- Exemple :

Algorithme *SimpleOuDouble*

{Cet algorithme saisit une valeur entière et affiche son double si cette donnée est inférieure à un seuil donné.}

constante (SEUIL : entier) ← 10

Variable val : entier

début

Afficher("Donnez-moi un entier :") { saisie de la valeur entière}

Saisir(val)

si val < SEUIL { comparaison avec le seuil}

alors **Afficher** ("Voici son double :", val × 2)

sinon **Afficher** ("Voici la valeur inchangée :", val)

fsi

fin

MAP - UNS

25

STRUCTURE ALTERNATIVE

« SI ... ALORS ... SINON ... FSI » (2)

- Ou instruction conditionnelle

si <expression logique>

alors instructions

[**sinon** instructions]

fsi

- Si l'expression logique (la condition) prend la valeur **vrai**, le premier bloc d'instructions est exécuté;
- si elle prend la valeur **faux**, le second bloc est exécuté (s'il est présent, sinon, rien).

MAP - UNS

26

STRUCTURE ALTERNATIVE

« SI ... ALORS ... SINON ... FSI » (3)

- Autre écriture de l'exemple :

Algorithme *SimpleOuDouble*

{Cet algorithme saisit une valeur entière et affiche son double si cette donnée est inférieure à un seuil donné.}

constante (SEUIL : entier) $\leftarrow 10$

Variable val : entier

début

Afficher ("Donnez-moi un entier : ") { saisie de la valeur entière }

Saisir(val)

si val < SEUIL { comparaison avec le seuil }

alors val \leftarrow val \times 2

Fsi

Afficher ("Voici la valeur val :", val)

fin

MAP - UNS

27

STRUCTURES ALTERNATIVES

IMBRIQUÉES

- **Problème:** afficher :
 - "Reçu avec mention Assez Bien " si une note est supérieure ou égale à 12,
 - " Reçu mention Passable" si elle est supérieure à 10 et inférieure à 12, et
 - "Insuffisant" dans tous les autres cas.

si note ≥ 12

alors afficher("Reçu avec mention AB")

sinon si note ≥ 10

alors afficher(« Reçu mention Passable")

sinon afficher("Insuffisant")

fsi

fsi

MAP - UNS

28

SELECTION CHOIX MULTIPLES « SELON » (1)

selon <identificateur>
 (liste de) valeur(s) : instructions
 (liste de) valeur(s) : instructions
 ...
 [**autres**: instructions]

- S'il y a plus de deux choix possibles, l'instruction **selon** permet une facilité d'écriture

MAP - UNS

29

SÉLECTION CHOIX MULTIPLES « SELON » (2)

selon abréviation
 "M" : afficher(" Monsieur ")
 "Mme" : afficher(" Madame ")
 "Mlle" : afficher(" Mademoiselle ")
autres : afficher(" Monsieur, Madame ")

Équivalent avec instruction Conditionnelle

```

si abréviation = "M "
  alors afficher( "Monsieur" )
  sinon si abréviation = « Mlle »
    alors afficher("Mademoiselle")
    sinon si abréviation = "Mme"
      alors afficher( "Madame" )
      sinon afficher( "Monsieur, Madame " )
    fsi
  fsi
fsi
  
```

MAP - UNS

30

SÉLECTION CHOIX MULTIPLES

EXEMPLE (3) AVEC INVERSION DES TESTS

selon abréviation

"M" : afficher(" Monsieur ")

"Mme" : afficher(" Madame ")

"Mlle" : afficher(" Mademoiselle ")

autres : afficher(" Monsieur, Madame ")

Équivalent avec instruction Conditionnelle

si abréviation = "Mme "

alors afficher(« Madame »)

sinon si abréviation = « Mlle »

alors afficher("Mademoiselle")

sinon si abréviation = "M"

alors afficher("Monsieur")

sinon afficher("Monsieur, Madame ")

fsi

fsi

fsi

MAP - UNS

31

SÉLECTION CHOIX MULTIPLES

EXEMPLE (4) AVEC SI ... ALORS ... FSI SÉQUENTIELS

selon abréviation

"M" : afficher(" Monsieur ")

"Mme" : afficher(" Madame ")

"Mlle" : afficher(" Mademoiselle ")

autres : afficher(" Monsieur, Madame ")

Équivalent avec instruction Conditionnelle

si abréviation = "Mme "

alors afficher(« Madame »)

fsi

si abréviation = « Mlle »

alors afficher("Mademoiselle")

fsi

si abréviation = "M"

alors afficher("Monsieur")

sinon afficher("Monsieur, Madame ")

fsi

MAP - UNS

32

TO DO

Calculez le nombre d'instructions nécessaires pour évaluer l'exécution dans le cas de 24 étudiants et 2 étudiantes célibataires.

Traiter les 3 cas de exemple 2, 3 et 4.

MAP - UNS

33

RÉPÉTITION D'UN TRAITEMENT BOUCLE « POUR »

- Exemple

Algorithme FaitLeTotal

{Cet algorithme fait la somme des *nbVal* données qu'il saisit}

variables *nbVal*, *cpt* : **entiers**
 valeur, *totalValeurs*: **réels**

début

 {initialisation du traitement}

afficher("Combien de valeurs voulez-vous saisir ?")

saisir(*nbVal*)

 {initialisation du total à 0 avant cumul}

totalValeurs ← 0

 {traitement qui se répète *nbVal* fois}

pour *cpt* ← 1 à *nbVal* **faire**

afficher("Donnez une valeur :")

saisir(*valeur*)

totalValeurs ← *totalValeurs* + *valeur* {cumul}

fpour

 {édition des résultats}

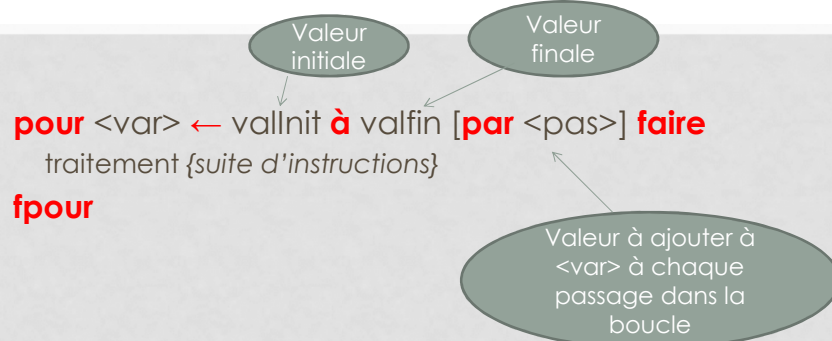
afficher("Le total des ", *nbVal*, "valeurs est ", *totalValeurs*)

fin

MAP - UNS

34

BOUCLE « POUR »



- **Fonction:** répéter une suite d'instructions un certain nombre de fois
- Pour utilisée **quand le nombre d'itération** est connu

MAP - UNS

35

SÉMANTIQUE BOUCLE « POUR »

- l'instruction **pour**:
 - initialise une variable de boucle (le compteur)
 - incrémente cette variable de la valeur de « pas »
 - vérifie que cette variable ne dépasse pas la borne supérieure
- **Attention :**
 - -le traitement ne doit pas modifier la variable de boucle

~~Pour cpt ← 1 à MAX faire
 si (...) alors
 cpt ← MAX
 fpour~~

INTERDIT !

MAP - UNS

36

RÉPÉTITION D'UN TRAITEMENT À NOMBRE ITÉRATIONS INCONNU

« TANT QUE ... FAIRE »

- Exemple

Algorithme FaitLeTotal

{Cet algorithme fait la somme des *nbVal* données qu'il saisit, arrêt à la lecture de -1 }

constante (STOP : entier) ← -1

variables val, totalValeurs: entiers

début

totalValeurs ← 0

afficher("Donnez une valeur,", STOP, " pour finir.") {**amorçage**}

saisir(val)

tant que val ≠ STOP **faire**

totalValeurs ← totalValeurs + val {**traitement**}

afficher("Donnez une autre valeur,", STOP, " pour finir.")

saisir(val) {**relance**}

ftq

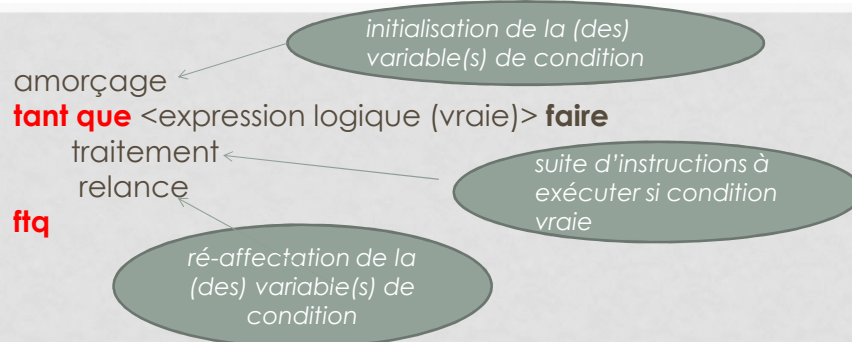
afficher("La somme des valeurs saisies est", totalValeurs)

fin

MAP - UNS

37

BOUCLE « TANT QUE ... FAIRE »



- Fonction:** répéter une suite d'instructions un certain nombre de fois

MAP - UNS

38

BOUCLE « TANT QUE ... FAIRE »

- Structure itérative "universelle"
 - n'importe quel contrôle d'itération peut se traduire par le "tant que"
- Structure itérative irremplaçable dès que la **condition d'itération** devient **complexe**

MAP - UNS

39

BOUCLE « TANT QUE ... FAIRE »

- Exemple:
 - saisir des valeurs, les traiter, et s'arrêter à la saisie de la valeur d'arrêt **-1** ou après avoir saisi **5** données.
- Constantes** (STOP : entier) ← -1
(MAX : entier) ← 5
- Variables** nbVal, val : entiers
- Début**
- nbVal ← 0 {compte les saisies **traitées**}
- saisir**(val) {saisie de la 1^{ère} donnée}
- tant que** val ≠ STOP **et** nbVal < MAX **faire**
- nbVal ← nbVal + 1 ... {traitement de la valeur saisie}
- saisir**(val) {relance}
- Ftq**
- afficher**(val, nbVal) {valeurs en sortie de boucle}
- **Remarque :** La valeur d'arrêt n'est jamais traitée (et donc, jamais comptabilisée)

MAP - UNS

40

BOUCLE « TANT QUE ... FAIRE »

- **Interpréter l'arrêt des itérations**

```

.....
nbVal←0 {compte les saisies traitées}
saisir(val) {saisie de la 1ère donnée}
tant que val ≠STOP et nbVal< MAX faire
    nbVal←nbVal+ 1...{traitement de la valeur saisie}
    saisir(val) {relance}
Ftq
si val = STOP
    alors {la dernière valeur testée était la valeur d'arrêt}
        afficher(«Sortie de boucle car saisie de la valeur d'arrêt »)
        {toutes les données significatives ont été traitées.}
    sinon {il y avait plus de 5 valeurs à tester}
        afficher(«Sortie de boucle car nombre maximum de valeurs à traiter
atteint ») { des données significatives n'ont pas pu être traitées.}
fsi

```

MAP - UNS

41

COMPARAISON BOUCLES « POUR » ET « TANT QUE » (1)

```

pour cpt ←1 à nbVal faire
    afficher("Donnez une valeur :")
    saisir(valeur)
    totalValeurs←totalValeurs+ valeur {cumul}
fpour

• Est équivalent à
cpt ←0
tant que cpt <nbVal faire
    afficher("Donnez une valeur :")
    saisir(valeur)
    totalValeurs←totalValeurs+ valeur {cumul}
    cpt ←cpt + 1 {compte le nombre de valeurs traitées}
ftq

```

MAP - UNS

42

COMPARAISON BOUCLES « POUR » ET « TANT QUE » (2)

- Implicitement, l'instruction **pour**:
 - initialise un compteur
 - incrémente le compteur à chaque pas
 - vérifie que le compteur ne dépasse pas la borne supérieure
- Explicitement, l'instruction **tant que** doit
 - initialiser un compteur {amorçage}
 - incrémenter le compteur à chaque pas {relance}
 - vérifier que le compteur ne dépasse pas la borne supérieure {test de boucle}

MAP - UNS

43

QUAND CHOISIR « POUR » OU « TANT QUE » ?

- Nombre d'itération connu à l'avance : **POUR**
 - Parcours de tableaux
 - Test sur un nombre donné de valeurs
- Boucle s'arrête sur événement particulier : **TANT QUE**
 - Itération avec arrêt décidé par saisie utilisateur

MAP - UNS

44

MAIS ON N'A PAS FINI D'ITÉRER !

- **Boucle « répéter ... tant que »** : exemple

Algorithme Essai

{Cet algorithme a besoin d'une valeur positive paire}

Variables valeur : entier

Début

Répéter

afficher("Donnez une valeur positive non nulle : ")

saisir(valeur)

tant que valeur ≤ 0

afficher("La valeur positive non nulle que vous avez saisie est ")

afficher(valeur)...{traitement de la valeur saisie}

fin

MAP - UNS

45

BOUCLE « RÉPÉTER ...TANT QUE »

Répéter

(ré)affectation de la (des) variable(s) de condition
traitement

Tant que <expression logique (vraie)>

- **Fonction:** exécuter une suite d'instructions **au moins une fois** et la répéter tant qu'une condition est remplie
- **Remarque:** le traitement dans l'exemple précédent se limite à la ré-affectation de la variable de condition (**saisir**(valeur))

MAP - UNS

46

COMPARAISON «RÉPÉTER» ET «TANT QUE»

Répéter

afficher("Donnez une valeur positive paire :")

saisir(valeur)

tant que(valeur < 0 **ou** (valeur % 2) ≠ 0)

- Équivaut à

afficher("Donnez une valeur positive paire :") **saisir**(valeur)

tant que(valeur < 0 **ou** (valeur % 2) ≠ 0) **faire**

afficher("Donnez une valeur positive paire:")

saisir(valeur)

ftq

MAP - UNS

47

COMPARAISON «RÉPÉTER» ET «TANT QUE»

- boucle **tant que**
 - condition vérifiée **avant** chaque exécution du traitement
 - le traitement peut donc ne pas être exécuté
 - de plus : la condition porte surtout sur la saisie de nouvelles variables (relance)
- boucle **répéter ... tant que**
 - condition vérifiée **après** chaque exécution du traitement
=>le traitement est **exécuté au moins une fois**
 - de plus: la condition porte surtout sur le résultat du traitement
- **Remarque:** la boucle répéter est typique pour les saisies avec vérification

MAP - UNS

48

DE L'ÉNONCÉ À LA BOUCLE

- saisir des données et s'arrêter **dès que leur somme dépasse 500**

somme \leftarrow 0

répéter

saisir(val)

somme \leftarrow somme + val

tant que somme \leq 500

saisir(val)

somme \leftarrow val

tant que somme \leq 500 **faire**

saisir(val)

somme \leftarrow somme + val

ftq

MAP - UNS

49

DE L'ÉNONCÉ À LA BOUCLE

- saisir des données et s'arrêter **dès que leur somme dépasse 500**

somme \leftarrow 0

répéter

saisir(val)

somme \leftarrow somme + val

tant que somme \leq 500

MAP - UNS

50