

Semestre Universitaire de Réorientation
Université Claude Bernard Lyon I
Année Universitaire 2018-2019

MODULE INFORMATIQUE S.U.R.

Elodie DESSEREE

Responsable Licence Informatique Lyon 1

elodie.desseree@univ-lyon1.fr





ENSEIGNEMENTS DU MODULE

- Environ 80 h d'enseignements en Algorithmique / programmation
 - CM / TD / TP intégrés.
 - 1 seule salle avec équipements spécifiques (vidéoprojecteur, ordinateurs TP)
- UE créditante de 6 ects
- Page web : <http://perso.univ-lyon1.fr/elodie.desseree/SUR/>



PROGRAMME DE L'UE

- Algorithmique
 - syntaxe algorithmique, écriture d'algorithmes
 - structures de contrôle : itérations, conditions
 - sous-programmes (fonctions / procédures)
 - mode de passage des paramètres dans des sous-programmes
 - tableaux / chaînes de caractères
 - structures
- Programmation impérative
 - Traduction dans un langage de programmation adapté des notions algorithmiques étudiées
 - alternative, séquence,
 - fonction / procédure,
 - structures, tableaux, chaînes de caractères, ...

PRÉAMBULE : QU'EST-CE QUE L'ALGO / PROG ?

- Génie Logiciel / Programmation
- Classification(s) des langages de programmation
- Le vocabulaire de base de la programmation
- Quelques exemples d'outils de programmation :
 - Logo / Lighthbot / Robozzle
 - Javascript
 - Codeblocks
 - Scratch





GÉNIE LOGICIEL

- Le **génie logiciel** (ou Software engineering en anglais) est un domaine de l'ingénierie qui permet la conception, la réalisation et la maintenance des systèmes **logiciels** de qualité :
 - Ensemble d'entités nécessaires au fonctionnement d'un processus de traitement automatique de l'information (Programmes, données, documentation...)
 - Ensemble de programmes qui permet à un système informatique d'assurer une tâche ou une fonction en particulier

Logiciel = programme + utilisation



SPÉCIFICATIONS DU LOGICIEL

- Spécification fonctionnelle
 - Fonctionnalités du logiciel, réponse aux besoins des utilisateurs
- Spécification non fonctionnelle
 - Facilité d'utilisation: prise en main et robustesse
 - Performance: temps de réponse, débit, fluidité...
 - Fiabilité: tolérance aux pannes
 - Sécurité: intégrité des données et protection des accès
 - Maintenabilité: facilité à corriger et à faire évoluer le logiciel
 - Portabilité: adaptabilité à d'autres environnements matériels ou logiciels

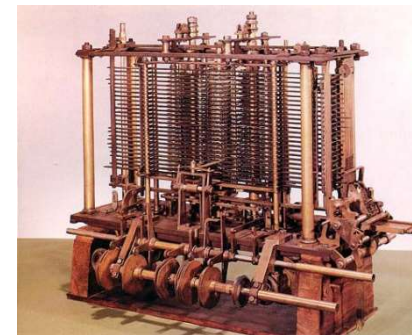


LA PROGRAMMATION

- ensemble des activités qui permettent l'écriture des programmes informatiques
- étape importante du développement de logiciels
- Utilisation d'un langage de programmation.
- rédaction du (ou des) code source d'un logiciel. On utilise plutôt le terme développement pour dénoter l'ensemble des activités liées à la création d'un logiciel et des programmes qui le composent

NAISSANCE DE LA PROGRAMMATION


- Première machine programmable : métier à tisser de Jacquard en **1801** (suite de cartons perforés avec le motif à reproduire lors du tissage). Repris par IBM bien plus tard !
- En **1936**, création de l'ordinateur programmable : la machine de Turing
 - premier calculateur universel programmable
 - invention des concepts et des termes de programmation et de programme.





UN ALGORITHME C'EST QUOI ?

- Un algorithme : suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat
- L'algorithmique intervient dans la vie de tous les jours
 - Une **recette** de cuisine
 - des entrées (les ingrédients, le matériel utilisé)
 - des instructions élémentaires simples
 - un résultat : le plat préparé
 - Le **tissage** automatisé par le métier Jacquard
 - Un **casse-tête** : Rubik's Cube.
 - En **sport** : exécution de séquences répondant à des finalités d'attaque, de défense, de progression.



CHOIX DU LANGAGE DE PROGRAMMATION

- Critère à prendre en considération :
 - Nature : typage fort ou faible
 - Portabilité : quel système d'exploitation ?
 - Stabilité : langage ancien ou récent ?
 - Pérennité : réussit-il à s'imposer comme un standard ou est-il exotique ?
 - Métier et maintenance : y a-t-il des gens compétents ? Est-il bien documenté ?
 - Temps d'apprentissage : pour devenir autonome
 - Temps de maîtrise : pour faire le tour du langage
 - Ouverture : interactions possibles avec d'autres langages ?
 - Sécurité : robustesse face aux attaques ?
 - Licence et droits : open source ? Licence payante ?
 - Scalabilité : capacité face à des utilisations simultanées



LA PROGRAMMATION POUR TOUS

- L'informatique, omniprésente dans les différentes sphères de l'entreprise, de la recherche, des services, de la culture et des loisirs
- Volonté politique :
 - plan informatique pour tous (IPT) lancé dès 1985
 - De plus en plus présent dans les programmes scolaires (généralisé depuis la rentrée 2012)
 - B2I niveau 1 : fin de cycle 3 à l'école élémentaire
 - B2I niveau 2 : au collège
 - B2I niveau 3 : au lycée
 - C2I niveau 1 : première année de licence STS
 - Savoir programmer permettrait de se faire embaucher plus facilement
- Multiplication des outils d'apprentissage depuis le plus jeune âge.



LANGAGES

- Classement des langages de programmation en fonction de leur utilisation :
http://www.tiobe.com/tiobe_index?page=index
- Plus de 700 langages de programmation !
- Quelques exemples très différents détaillés
 - Les standards
 - Ceux pour apprendre dès le plus jeune age
 - Ceux pour dessiner
 - ...



LOGO

- Public visé : enfants à partir de 4 ans
- Tortue exécutant principalement des opérations de dessin.
- <http://tortue-logo.fr/fr/tortue-logo>
- **Commandes de base**
 - **av N** : Avance de N pixels
 - **td N** : tourne à droite de N degrés
 - **tg N** : tourne à gauche de N degrés
 - **re N** : recule de N pixels
 - **lc** : lève le crayon
 - **bc** : baisse le crayon
 - **ct** : cache la tortue
 - **mt** : montre la tortue
 - **ve** : efface l'écran et revient à la position initiale
 - **nettoie** : efface l'écran sans modifier la position
 - **repete N []** : répète N fois la liste d'actions entre crochets
- Exercice dessiner un carré de coté 100 !



LIGHTBOT

- Public visé : enfants à partir de 4 ans
- Petit robot sachant avancer, tourner, sauter et allumer des lumières.
- Multi plateforme : tablettes (mac et android), PC, ...
- Très guidé et intuitif : apprendre à programmer en s'amusant
- <http://lightbot.com/flash.html>



JAVASCRIPT

- Langage de scripts principalement employé dans les pages web interactives
- JavaScript sert généralement à contrôler les données saisies dans des formulaires HTML, ou à interagir avec le document HTML
- Du code JavaScript peut être intégré directement au sein des pages web, pour y être exécuté sur le poste client. Le navigateur Web prend en charge l'exécution de ces programmes appelés scripts.
- <http://js.do/>
- Exercice : affichez le périmètre d'un cercle dont le rayon est demandé à l'utilisateur en utilisant les deux instructions suivantes :
 - `alert(Chaîne à afficher à l'écran)`
 - `variable=prompt(Question à afficher à l'écran)`



SCRATCH

- Scratch est une implémentation visuelle et dynamique du langage de programmation Smalltalk basé sur Squeak
- Version online
(https://scratch.mit.edu/projects/editor/?tip_bar=getStarted)
- Version offline à télécharger : <https://scratch.mit.edu/download>
- Scratch est constitué de multiples briques permettant d'exécuter une action précise. Il existe 10 catégories de briques différentes (classées par couleurs)
- Exercices
 - Dessiner un carré de coté 100
 - Afficher le périmètre d'un cercle dont le rayon est demandé à l'utilisateur



LANGAGE C/ C++

- langage de programmation impératif et généraliste
- inventé au début des années 1970 pour réécrire UNIX, devenu un des langages les plus utilisés.
- de nombreux langages plus modernes comme C++, Java et PHP reprennent des aspects de C.
- langage compilé (ligne de commande ou utilisation d'un IDE)
 - Codeblocks
 - Visual studio
 - Devcpp
 - ...
- nombreuses bibliothèques : graphique / scientifique ...
- Langage de programmation largement utilisé en licence informatique



CONCLUSION

- Aujourd'hui découverte
 - Des différentes notions d'algorithme, de programme
 - Quelques petites expériences de programmation
 - Divers langages
 - Diverses applications
- Dans la suite du cours
 - Apprendre à écrire des algorithmes
 - Programmer en langage C/C++



BASES DE L'ALGORITHMIQUE





PLAN

- Le programme de l'UE, objectifs
- Le fonctionnement interne d'un ordinateur
- La programmation
- Le langage algorithmique
- La syntaxe algorithmique
- Quelques exemples complets



PROGRAMME DE L' UE

- Algorithmique :
 - syntaxe algorithmique, écriture d'algorithmes
 - structures de contrôle : itérations, conditions
 - sous-programmes (fonctions / procédures)
 - mode de passage des paramètres dans des sous-programmes
 - tableaux / chaînes de caractères
 - structures
- Programmation impérative :
 - Traduction dans un langage de programmation adapté des notions algorithmiques étudiées (fonction / procédure, alternative, séquence, structures, tableaux, chaînes de caractères, ...)



OBJECTIFS DE L' UE

- Analyser un problème
- Le formaliser
- Concevoir une solution (algorithme)
- Programmer l'algorithme
- Exécuter le programme sur un ordinateur

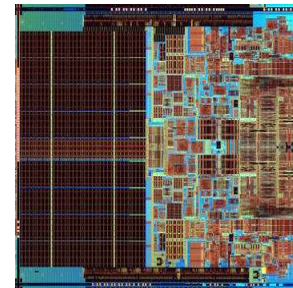
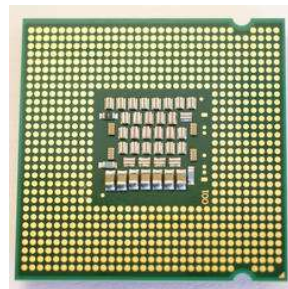


PLAN

- Le programme de l'UE
- **Le fonctionnement interne d'un ordinateur**
- La programmation
- Le langage algorithmique
- La syntaxe algorithmique
- Quelques exemples complets

COMPOSITION D'UN ORDINATEUR

- Vision **simpliste** du contenu d'un ordinateur
 - Processeur : effectue les opérations
 - Mémoire(s), disques : stockage données, instructions
 - ...
- Effectue des opérations à partir de données
- Vues d'un processeur



LE PROCESSEUR COMPREND :

- Programme (séquence d'instructions du processeur)

| | |
|------------------------|--------------------|
| cc2: 55 | push %ebp |
| cc3: 89 e5 | mov %esp,%ebp |
| cc5: 53 | push %ebx |
| cc6: 83 ec 14 | sub \$0x14,%esp |
| cc9: e8 fc ff ff ff | call cca |
| cce: 81 c3 02 00 00 00 | add \$0x2,%ebx |
| cd4: 8b 45 08 | mov 0x8(%ebp),%eax |
| cd7: 89 44 24 04 | mov %eax,0x4(%esp) |
| cdb: 8b 45 08 | mov 0x8(%ebp),%eax |
| cde: 89 04 24 | mov %eax,(%esp) |

Code machine

Assembleur

- Seul langage compris par le processeur
- Codage hexadécimal des instructions
→ Quasi inutilisable pour programmeur



PLAN

- LIFAP1 : programme de l'UE
- LIFAP1 / Culture Numérique / Autres UE informatiques
- Le fonctionnement interne d'un ordinateur
- **La programmation**
- Le langage algorithmique
- La syntaxe algorithmique
- Quelques exemples complets



POURQUOI PROGRAMMER ?

- Programmation existe partout
 - Réveil
 - Digicode
 - Téléphone
 - Tablette ...
- Besoin d'effectuer des nouvelles tâches
 - ➔ besoin d'écrire des programmes nouveaux
 - Par non informaticien : formalisation en français
 - Par informaticien : langage compréhensible par lui et la machine



UN PROGRAMME C'EST QUOI ?

- Un programme, c'est tout ce qui fonctionne sur votre ordinateur, par exemple :
 - Un jeu vidéo (FIFA2016, AngryBrids, ...)
 - Un lecteur vidéo (comme VLC par exemple ou Youtube),
 - Ou même un truc tout simple comme OpenOffice, Mozilla Firefox.
 - Et le plus important **le système d'exploitation** (Windows, Android ...)
- **Sans programme pas d'application sur votre ordinateur !**

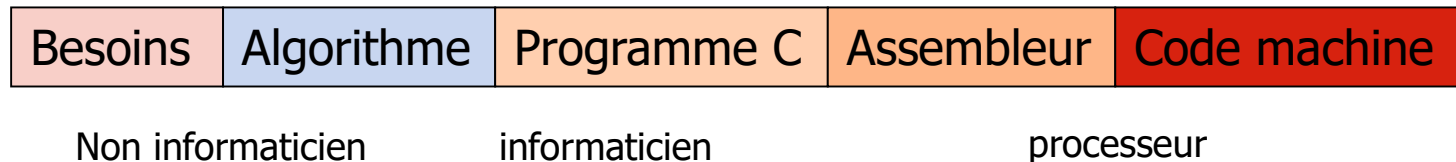


LE LANGAGE DE PROGRAMMATION

- Langage commun entre
 - Le programmeur
 - Le processeur : traduit en assembleur puis en code machine
- Grande diversité
 - Langage C/C++ (ce semestre)
 - Python (ISN au lycée pour certains)
 - Scheme (fonctionnel et récursif)
 - Java, Matlab, Mathematica, macros word / excel (écrites en Visual Basic for Applications VBA)...
 - ...
 - Plus de 700 langages de programmation !!

DU PROBLÈME AU PROGRAMME

- Besoins exprimés en français (cahier des charges)
- Traduction dans un langage "universel"
= algorithmique intermédiaire
- Traduction de l'algorithme en programme C
- Puis en code assembleur
- Puis en code machine compréhensible par le processeur





PLAN

- LIFAP1 : programme de l'UE
- LIFAP1 / Culture Numérique / Autres UE informatiques
- Le fonctionnement interne d'un ordinateur
- La programmation
- **Le langage algorithmique**
- La syntaxe algorithmique
- Quelques exemples complets



ALGORITHME : DÉFINITION

- Un algorithme est une méthode
 - Suffisamment générale pour permettre de traiter toute une classe de problèmes
 - Combinant des opérations suffisamment simples pour être effectuées par une machine
- Pour un problème donné, il peut y avoir plusieurs algorithmes ou aucun



ALGORITHME : PROPRIÉTÉS

- **Lisible** : l'algorithme doit être compréhensible même par un non-informaticien
- **Haut niveau** : doit pouvoir être traduit en n'importe quel langage de programmation → ne pas faire appel à des notions techniques relatives à un programme particulier ou bien à un système d'exploitation donné
- **Précis / non ambigu** : chaque élément de l'algorithme ne doit pas porter à confusion
- **Concis** : ne doit pas dépasser une page, sinon décomposer le problème en plusieurs sous-problèmes
- **Structuré** : un algorithme doit être composé de différentes parties facilement identifiables



ALGORITHME : MÉTHODOLOGIE

- Trois étapes caractérisent la résolution d'un problème
 1. **comprendre la nature du problème** posé
et préciser les **données** fournies
("entrées" ou **"input"** en anglais)
 2. **préciser les résultats** que l'on désire obtenir
("sorties" ou **"output"** en anglais)
 3. **déterminer le processus de transformation**
des données en résultats.
- Ces trois étapes ne sont pas indépendantes.



ALGORITHMIQUE / LANGAGE PROGRAMMATION

- Un algorithme est
 - Une **suite d'instructions élémentaires** décrites dans un langage universel exécutées de manière **séquentielle**
 - Indépendant du langage de programmation
- Un langage de programmation
 - Est un langage commun entre machine et programmeur
 - Implante ou réalise un algorithme



PLAN

- LIFAP1 : programme de l'UE
- LIFAP1 / Culture Numérique / Autres UE informatiques
- Le fonctionnement interne d'un ordinateur
- La programmation
- Le langage algorithmique
- **La syntaxe algorithmique**
- Quelques exemples complets

L' INSTRUCTION, LA SÉQUENCE

- Instruction :
 - Étape dans un programme informatique
 - Dicte à l'ordinateur l'action nécessaire qu'il doit effectuer avant de passer à l'instruction suivante.
 - Opération élémentaire
 - Comprise et exécutée par le processeur
- Séquence / suite d'instructions
 - Suite bloquée d'instructions qui sont exécutées dans l'ordre où elles sont écrites, dans toutes les circonstances du traitement.
 - Délimitée par **Début** et **Fin** (→ **bloc**)

Début

instruction1

instruction2

...

instructionN

Fin



LA VARIABLE / LA CONSTANTE

- Une **variable**

- nom utilisé dans un programme pour faire référence à une donnée manipulée par programme
- peut contenir un entier, un réel, un caractère...
- associe un nom ou symbole à une valeur
- sa valeur peut éventuellement varier au cours du temps

- Une **constante**

- nom utilisé pour faire référence à une valeur permanente (dont la valeur ne change pas au cours du programme).
 - $\text{PI} = 3.14159\dots$



LE TYPE DES DONNÉES

- Définit les valeurs que peut prendre une donnée, ainsi que les opérateurs qui peuvent lui être appliqués
- Types utilisés en algorithmique :
 - Caractère : 'c' , 'a' , '-' , '!' ...
 - Entier : 3 0 -3 -789
 - Réel : 0 3,345 -7,678
 - Booléen : VRAI / FAUX
 - ...



LA DÉCLARATION DES VARIABLES

- La *déclaration* permet de donner un **nom** à la variable
 - Eventuellement de lui associer un **type**,
 - Ainsi qu'une **valeur initiale**.
- Exemples
 - indice : entier
permettra de déclarer une variable "indice" de type entier
 - Est_majuscule : booléen
permettra de déclarer une variable booléenne
- La variable doit avoir un nom aussi évocateur que possible de son contenu



L'AFFEKTATION

- Attribue une valeur à une **variable**
- Symbolisée en algorithmique par le symbole "**←**"

- La valeur peut être
 - le résultat d'une expression

variable ← expression

var1 ← a + 2*racine(15)

- une valeur numérique

a ← 2 (la variable *a* contient la valeur 2)



OPÉRATIONS SUR LES VARIABLES

- Affectation : variable \leftarrow expression
- La variable contient la valeur de l'expression
- Cette valeur est conservée jusqu'à la prochaine affectation
- Une variable peut apparaître dans une expression, elle sera remplacée par la valeur qu'elle contient au moment du calcul de l'expression



CONTENU D'UNE VARIABLE

- Pour pouvoir stocker la valeur et vérifier qu'une variable est correctement utilisée,
 - une variable a un type
- Un type est :
 - un domaine de valeurs (ensemble des valeurs possibles)
 - Entiers, réels
 - Booléen
 - caractères
 - un ensemble d'opérations pour manipuler ces valeurs
 - Addition, soustraction, multiplication ...
 - Opérations logiques
 - Concaténation, substitution ...



EXEMPLE 1 : CALCUL DU PRODUIT

- On veut calculer le produit de a par b et stocker le résultat dans une variable C

- En algorithmique on écrira :

Début

a,b,c : réels

déclaration des variables

$c \leftarrow a*b$

stockage du résultat

du calcul $a*b$ dans la variable c

Fin



LES ENTRÉES / SORTIES

- Assurent la communication programmeur / machine
- Données du problème (utilisateur → machine)
Lire (valeur) ou **Saisir** (Valeur)
- Résultats affichés à l'écran (machine → utilisateur)
Afficher (valeur) ou **Ecrire** (Valeur)

LA CONDITIONNELLE

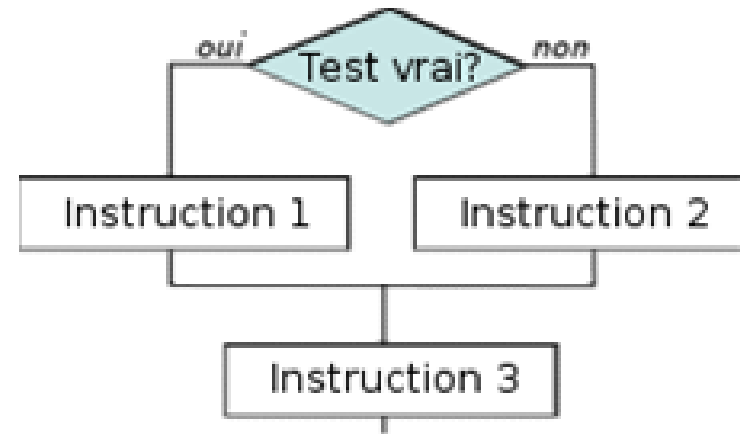
Si condition **alors**

instruction(s)

Sinon

instruction(s)

FinSi



- Condition = expression booléenne (vrai / faux)
 - Élémentaire
 - Complexe (conjonction, négation ou disjonction de conditions élémentaires et/ou complexes)
- Partie sinon facultative : il n'y a pas nécessairement de traitement à effectuer.

LA CONDITIONNELLE : EXEMPLES

○ Exemple 1 sans "sinon"

Si (A>2)

Alors B←A*3

FinSi

○ Exemple 2 avec "sinon"

Si ((A<10) **et** (B>racine(A*5)))

Alors

B←A*3

A← A+B

Sinon

A←A+2

B←A*B

FinSi

Les opérateurs de comparaison sont :

- égal à...
- différent de...
- strictement plus petit que...
- strictement plus grand que...
- plus petit ou égal à...
- plus grand ou égal à...

Expression booléenne : 3 opérateurs logiques :
ET, OU, NON



STRUCTURE CONDITIONNELLE SELON

- aussi appelée à **choix multiple** ou **sélective**
- sélectionne entre plusieurs choix à la fois, et non entre deux choix alternatifs (le cas de la structure SI).

- **Syntaxe**

SELON (sélecteur) FAIRE

Cas <liste de valeurs-1> : <suite d'action (s)-1>

[Cas <liste de valeur-2> : <suite d'action (s)-2>

.....]

[Autrement : <suite d'action (s)-n>]

FINSELON

- Le **sélecteur** est une variable de type entier ou caractère



STRUCTURE CONDITIONNELLE SELON : EXEMPLE

- Afficher la couleur en fonction d'un entier = 1 : rouge, 2 : orangé, 3 : jaune, 4 : vert, 5 : bleu, 6 : indigo et 7 : violet.

Selon couleur Faire

1 : afficher(" rouge")

2 : afficher(" orangé")

3 : afficher(" jaune")

4 : afficher(" vert")

5 : afficher(" bleu")

6 : afficher(" indigo ")

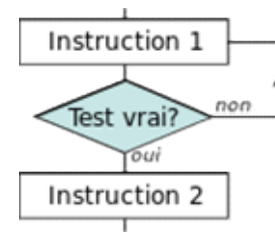
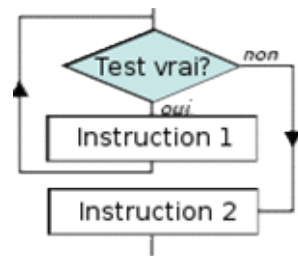
7 : afficher(" violet")

Autrement : afficher ("Couleur inconnue")

Fin selon

L'ITÉRATION

- Une **boucle** ou **itération** est une structure de contrôle destinée à exécuter une portion de code plusieurs fois de suite
- Les langages proposent en général plusieurs types de boucles :
 - boucle à pré-condition : la condition est vérifiée avant la première boucle
 - boucle à post-condition : la condition est vérifiée après la première boucle
 - boucle à condition d'arrêt : la condition est vérifiée au milieu de la boucle
 - boucle itérative : un compteur est utilisé pour compter le nombre d'itérations
 - boucle de parcours : la boucle est exécutée sur chacun des éléments d'une liste





ITÉRATION : BOUCLE CONDITIONNELLE

- Permet de réitérer une instruction ou une suite d'instructions jusqu'à ce qu'une condition ne soit plus vraie
- Condition évaluée **avant** d'effectuer les instructions

TantQue condition **faire**
instruction(s)

FinTantQue



BOUCLE CONDITIONNELLE : EXEMPLE

$i \leftarrow 1$

TantQue $i < 10$ faire

$a \leftarrow a * i$

$i \leftarrow i + 1$

FinTantQue

- Instruction qui modifie la condition
pour éviter les boucles infinies



ITÉRATIVE : BOUCLE CONDITIONNELLE

- Autre construction
- Condition évaluée **après** avoir effectué les instructions

Faire

instruction(s)

TantQue condition

Les instructions sont effectuées au moins une fois



BOUCLE CONDITIONNELLE : EXEMPLE

$i \leftarrow 1$

Faire

$i \leftarrow i+1$

Tant que $i < 10$

- instruction qui modifie la condition pour éviter les boucles infinies



BOUCLE INCONDITIONNELLE : POUR

- Cas particulier du TantQue

Pour compteur allant de ... à ... par pas de ...
faire

instruction(s)

FinPour

- Permet de répéter un nombre connu de fois
une suite d'instructions



BOUCLE INCONDITIONNELLE : EXEMPLES

- Compter de 1 à 10 (**incrémentation**)

Pour i allant de 1 à 10 par pas de 1 faire

$a \leftarrow i$

FinPour

- Compter de 10 à 1 (**décrémentation**)

Pour i allant de 10 à 1 par pas de -1 faire

$a \leftarrow i$

FinPour

- Compter de **deux en deux**

Pour i allant de 0 à 10 par pas de 2 faire

$a \leftarrow i$

FinPour

LA CONDITION

- Apparaît dans les "Si" et les "Tant Que"
- Variable booléenne qui renvoie comme valeur **VRAI** ou **FAUX**
- Combinaison de conditions :
conjonction (**ET**), disjonction (**OU**), négation (**NON**)
- Tables de vérité

| X | Y | X et Y |
|---|---|--------|
| V | V | V |
| V | F | F |
| F | V | F |
| F | F | F |


| X | Y | X ou Y |
|---|---|--------|
| V | V | V |
| V | F | V |
| F | V | V |
| F | F | F |

| X | Non X |
|---|-------|
| V | F |
| F | V |



PLAN

- LIFAP1 : programme de l'UE
- LIFAP1 / Culture Numérique / Autres UE informatiques
- Le fonctionnement interne d'un ordinateur
- La programmation
- Le langage algorithmique
- La syntaxe algorithmique
- Quelques exemples complets



EXEMPLE 2 : CALCUL DU PRODUIT

- Dans cet algorithme, on n'utilisera pas la multiplication !!

- Raisonnement : $5 * 4 = \underbrace{5 + 5 + 5 + 5}_{4 \text{ fois}}$

- Généralisation : $a * b = a + a + \dots + a$ (b fois)

- Formalisation :
tant qu'on n'a pas ajouté **b** fois **a**,
on ajoute **a** à la somme



EXEMPLE 2 : CALCUL DU PRODUIT

- Programme "complet" tel que vous aurez à les écrire dans le TD1.
- Traduction algorithmique avec un *tant que*

Début

a, b, somme : entiers

afficher ("donnez a et b")

lire (a)

lire (b)

somme \leftarrow 0

TantQue b \neq 0 **faire**


 somme \leftarrow somme + a

 b \leftarrow b - 1

FinTantQue

Afficher (somme)

Fin



EXEMPLE 2 : CALCUL DU PRODUIT

- Traduction algorithmique avec un "pour"

Début

a, b, somme : entiers /*mise en commentaires*/

somme ← 0 /* initialisation de la somme à 0 */

afficher ("donnez a et b")

lire (a)

lire (b)

Pour i allant de 1 à b par pas de 1 faire

 somme ← somme + a

FinPour

Afficher (somme)

Fin



EXEMPLE 3 : MINIMUM

- Détermination de la plus petite de 2 valeurs données par l'utilisateur
- Étapes de l'algorithme
 - Déclarer les variables à utiliser
 - Demander et saisir les valeurs de l'utilisateur
 - Comparer les deux valeurs
 - Utilisation d'une conditionnelle SI
 - Afficher la plus petite des deux



EXEMPLE 3 : MINIMUM

Début

a,b : entiers

afficher ("donnez la valeur de a")

lire (a)

afficher ("donnez la valeur de b")

lire (b)

si (a < b) **alors** afficher (a "est la plus
petite")

sinon afficher (b "est la plus petite")

fin si

Fin

La partie "sinon" équivaut à la condition $a \geq b$



EXEMPLE 4 : MINIMUM

Cette fois-ci on s'intéresse aussi au cas où les deux valeurs sont égales
(ni $a < b$ ni $b < a$)

→ 2 "si" imbriqués !!!

Début

a,b : entier

afficher ("donnez la valeur de a")

lire (a)

afficher ("donnez la valeur de b")

lire (b)

si ($a < b$) **alors** afficher (a "est la plus petite")

sinon **si** ($b < a$) **alors afficher** (b "est la plus petite")

sinon afficher ("les 2 valeurs sont égales")

fin si

fin si


Fin

La dernière partie "sinon" équivaut à la condition $a = b$



Exemple 5 : ON CONTINUE L' IMBRICATION

```
Si (val=1) alors afficher("Lundi")
  sinon si (val =2) alors afficher("Mardi")
    sinon si (val=3) alors afficher("Mercredi")
      sinon si (val=4) alors afficher("Jeudi")
        sinon si (val=5) alors afficher("Vendredi")
          sinon si (val=6) alors afficher("Samedi")
            sinon si (val=7) alors afficher("Dimanche")
              sinon afficher("Erreur")
            fin si
          fin si
        fin si
      fin si
    fin si
  fin si
fin si
```



EXEMPLE 5 : CHOIX MULTIPLE (SELON)

selon jour **faire**

1 : afficher("Lundi")

2 : afficher("Mardi")

3 : afficher("Mercredi")

4 : afficher("Jeudi")

5 : afficher("Vendredi")

6 : afficher("Samedi")

7 : afficher("Dimanche")

autrement : afficher("Erreur")

fin selon



CONCLUSION

- Tour d'horizon des notions de bases de l'algorithmique
 - Variable : déclaration, type
 - Instruction : séquence, bloc
 - Structures de contrôle
 - Conditionnelles : SI ... ALORS ... SINON ...
 - Boucles : TANT QUE, POUR
- Exemples simples d'applications