Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

ynov
CAMPUS

ALGORITHMIQUE

ALGORITHMIQUE

DÉFINITION : ALGORITHMIQUE

« C'est la logique d'écrire des algorithmes. »

ALGORITHME

DÉFINITIONS : ALGORITHME

« Ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé d
moyen d'un nombre fini d'opérations. » Larousse

« Ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations. » - Larousse

« Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat » - Wikipédia

« Un algorithme est une suite d'instructions détaillées qui, si elles sont correctement exécutées, conduit à un résultat donné. » - Mathématique

PIZZA FACILE, 45 MIN, 5 PERSONNES

Ingrédients

1 pâte à pizza prête à cuire
1 petite boîte de tomate
100 g de lardons nature
1 petite boîte de champignon de Paris en lamelles
2 poignées de gruyère râpée

Etape 1

Faire cuire dans une poêle les lardons et les champignons.

Etape 2

Dans un bol, verser la boîte de concentré de tomate, y ajouter un demi verre d'eau, ensuite mettre un carré de sucre
(pour enlever l'acidité de la tomate) une pincée de sel, de poivre, et une pincée d'herbe de Provence.

Etape 3

Dérouler la pâte à pizza sur le lèche frite de votre four.

Etape 4

Avec une cuillère à soupe, étaler délicatement la sauce tomate, ensuite y ajouter les lardons et les champignons bien dorer.
Parsemer de fromage râpée.

Etape 5

Mettre au four à 220°, thermostat 7-8, pendant 20 min (ou lorsque le dessus de la pizza est doré).

marmiton

PIZZA AU PESTO D'ÉPINARDS ET AU SAUMON FUMÉ

Ingrédients

4 galettes au blé
1 petit pot de pesto d'épinards
1 tomates coupée en dés
100 g de saumon fumé
80 g de mozzarella di buffala
poivre
thym

Préparation

Préchauffer le four thermostat 6 à 7. Superposer 2 galettes Sur la plaque du four recouverte de papier sulfurisé. Badigeonner chaque galette généreusement de pesto d'épinards. Parsemer de saumon fumé en lamelles, de tomate en dés, de mozzarella en petits morceaux, de poivre et de thym. Enfourner 10 minutes. Servir chaud avec une petite salade.

LE JOURNAL DES FEMMES

FULL ENGLISH PIZZA

Ingrédients

500g pack bread mix
a little sunflower oil
6 tbsp passata
4 pork sausages, skinned and quartered
140g mushooms, sliced
8 steaky bacon rashers, halved
4 edium eggs

Method

1. Heat oven to 220C/200C fan/gas 7. Make up the bread mix following pack instructions. On a lightly floured surface, roll out to fit a lightly oiled 30 x 40cm baking tray, or two smaller trays. Spread the passata over the base, dot over the sausages and mushrooms, add the bacon, then cook for 20 mins at the top of the oven.
2. Remove the pizza from the oven and crack the eggs on. Return to the oven and cook for 5 mins more, or longer depending how well-cooked you like your eggs.

VOS REACTIONS ?

VISION BOITE NOIRE

PROBLEME

MÉTHODE DE RÉSOLUTION D'UN PROBLÈME

1. Comprendre l'énoncé du problème

2. Décomposer le problème en sous-problèmes plus simple à résoudre

3. Associer à chaque sous problème, une spécification :
   a) Les données nécessaires
   b) Les données résultantes
   c) La démarche à suivre pour arriver au résultat en partant d'un ensemble de données.

4. Elaboration d'un algorithme.

HAMBURGER 100% MAISON

1. Comprendre l'énoncé du problème

2. Décomposer le problème en sous-problèmes plus simple à résoudre

3. Associer à chaque sous problème, une spécification :
   a) Les données nécessaires
   b) Les données résultantes
   c) La démarche à suivre pour arriver au résultat en partant d'un ensemble de données.

4. Elaboration d'un algorithme.

FORMALISER L'ÉCRITURE DE L'ALGORITHME

- ✓ Un langage commun
- ✓ Proche du langage naturel
- ✓ Indépendant de tout langage
- ✓ Facilement convertible en n'importe quelle autre langage
- ✓ Structuré

LANGAGE

UN LANGAGE

- ✓ **Règles de syntaxe**
  Un grammaire formelle, ces règles régissent les différentes manières dont les éléments du langage peuvent être combiné
- ✓ **Vocabulaire**
  L'ensemble des instructions construites d'après des symboles
- ✓ **Sémantique**
  Le sens de chacune des phrases qui peuvent être construites dans le langage
- ✓ **Alphabet**
  Lettre de A à Z par exemple

UN LANGAGE DE PROGRAMMATION

- ✓ **Règles de syntaxe**
  Un grammaire formelle, ces règles régissent les différentes manières dont les éléments du langage peuvent être combiné
- ✓ **Vocabulaire**
  L'ensemble des instructions construites d'après des symboles
- ✓ **Sémantique**
  Le sens de chacune des phrases qui peuvent être construites dans le langage
- ✓ **Alphabet**
  Lettre de A à Z par exemple
- ✓ **Commentaires**
  Texte permettant de donner des explications
- ✓ **Identifiants**
  Mot clefs permettant d'organiser le script.

FORMALISER L'ÉCRITURE DE L'ALGORITHME

ADL

Algorithm Definition Language

- ✓ Un langage commun
- ✓ Proche du langage naturel
- ✓ Indépendant de tout langage
- ✓ Facilement convertible en n'importe quelle autre langage
- ✓ Structuré

EXERCICES DE GROUPE

Présentation et comparatif des langages de programmation
11/10/2018

- ✓ Groupe de 3 personnes
- ✓ Choisir 3 langages
- ✓ Brève description des langages
- ✓ Quels est son but ? Ou quel était sont but à sa création?
- ✓ Les éléments de recherches :
  - ✓ Niveau ?
  - ✓ Typage ?
  - ✓ Paradigme ?
  - ✓ Comment le langage s'exécute?
- ✓ Format PPT

ALGORITHM DEFINITION LANGUAGE (ADL)

Pseudo-code

Logigramme

ALGORITHM DEFINITION LANGUAGE (ADL)

Pseudo-code

Logigramme

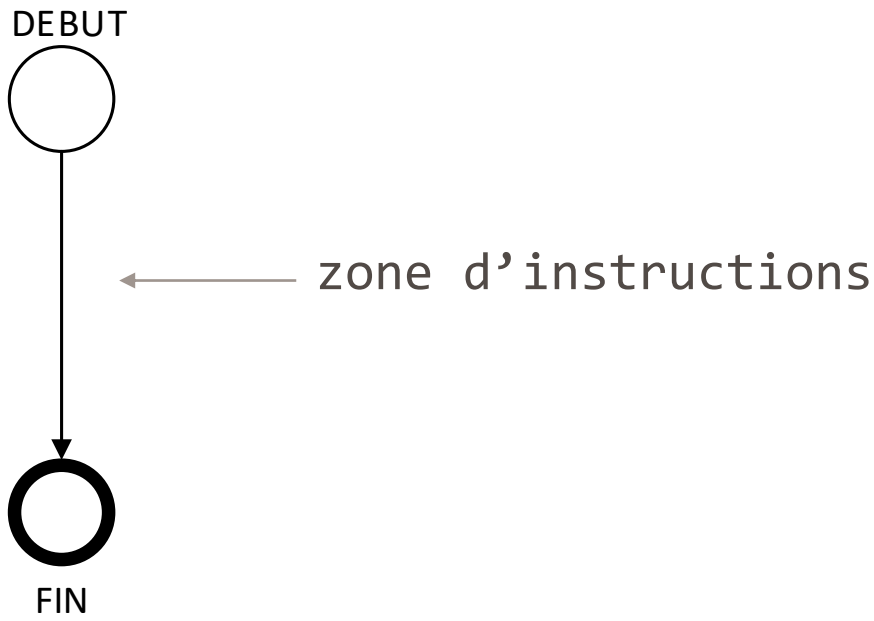
STRUCTURE DE L'ALGORITHME

Pseudo-code

```
ALGORITHME identifiant
    <zone de déclaration>
DEBUT
    <zone d'instructions>
FIN
```

/!\ indentation

Logigramme



/!\ pas de zone de déclaration

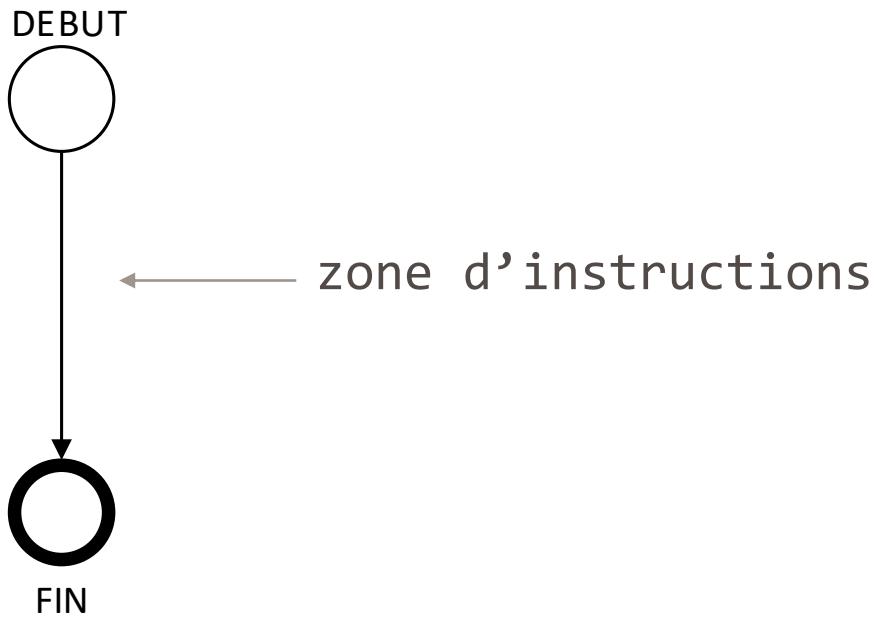
STRUCTURE DE L'ALGORITHME

Pseudo-code

```
ALGORITHME identifiant
    <zone de déclaration>
DEBUT
    <zone d'instructions>
FIN
```

/!\ indentation

Logigramme



/!\ pas de zone de déclaration

LES COMMENTAIRES

- ✓ **Commentaire**
  - Portion de code ignoré lors de l'exécution
  - A destination des développeurs
  - Permet d'expliquer une portion de code

```
ALGORITHME identifiant

DEBUT

    // Commentaire
    /*
       bloc
    */

FIN
```

ZONE DE DÉCLARATION

- ✓ **Variables**
  Repéré par son nom, pouvant contenir des données, qui pourront être modifiées lors de l'exécution du programme

- ✓ **Constantes**
  Une constante est une variable dont la valeur est inchangeable lors de l'exécution d'un programme

- ❖ Déclaré avec :
  Un identifiant.
  Un type : *integer, float, boolean, char, string*

- ❖ Symbole d'affectation : ←

```
ALGORITHME Exemple

CONSTANTE
 Type identifiant ← valeur

VARIABLE
 Type identifiant

DEBUT

    <zone d'instructions>

FIN
```

LES IDENTIFIANTS

- ✓ **Règles de construction d'un identifiant**
  - Lettres
  - Chiffres
  - Underscore : _
  - Ne doit jamais commencer par un chiffre.

- ✓ **Conseil**
  - L'identifiant doit représenter ce pourquoi on l'utilise : l'intention

toto	xy2	Clément	2nis
x	_TEST	3_TOTO	ELECTr0de
monNom	coucou	TeSt43_	Ingesup

LES TYPES

INTEGER	FLOAT	BOOLEAN	CHAR	STRING
35	3,14	'C'	'@'	
"COUCOU"	CoCOU	VRAI	"FAUX"	
'3'	K	TeSt43_	"23"	

ZONE D'INSTRUCTIONS

- ✓ **Affectations**
- ✓ **Calculs**
- ✓ **Lire, Ecrire, Afficher**
- ✓ **Appeler des fonctions**
- ✓ **Utiliser des structures de contrôle**
  - **Structures conditionnelles**
  - **Structures de répétitions**
- ✓ **Les data-structures**

AFFECTATION

✓ **Syntaxe :**

identifiant ⟵ valeur

```
ALGORITHME Exemple
VARIABLE
 INTEGER x, y, z

DEBUT
DEBUT
     x ⟵ 2
     y ⟵ 35
     z ⟵ 8
     z ⟵ x
     y ⟵ z


FIN
```

CALCULS

- **Les opérateurs :**
  - Addition : +
  - Soustraction : -
  - Multiplication : x
  - Division : div ou /
  - Modulo : mod
  - NON : NOT ou NON
  - OU : OR ou OU
  - ET : AND ou ET
  - OU ex : XOR ou OUEX

```
ALGORITHME Exemple
VARIABLE


DEBUT


FIN
```

LIRE, ECRIRE

```
ALGORITHME Exemple
VARIABLE
 INTEGER : age

DEBUT
    ECRIRE("Entrez votre age svp.")
    LIRE(age)
    AFFICHER("Votre age est : ", age)
FIN
```

LES PROCEDURE

Un sous programme, sous algorithme, ayant la capacité d'effectuer une tâche indépendante

PROCEDURE

```
PROCEDURE MaProcedure (type param1, type param2, …)
        <partie déclaration>
DEBUT
        <instructions>
FIN
```

LES FONCTIONS

Un sous programme, sous algorithme ayant la capacité d'effectuer une tâche indépendante

En math : y = f(x)

LES FONCTIONS

Un sous programme, ayant la capacité d'effectuer une tâche indépendante

En math : $y = f(x)$

LES FONCTIONS

Un sous programme, ayant la capacité d'effectuer une tâche indépendante

En math : $y = f(x)$

Une fonction peut avoir des paramètres

Une fonction peut retourner une valeur

FONCTION

```
FONCTION CalculerAge(ENTIER jour, ENTIER mois, ENTIER ANNEE): ENTIER
    ENTIER Val

DEBUT

    <zone d'instructions>

    RETOURNE val

FIN
```

FONCTION

```
ALGORITHME Exemple
VARIABLE

 INTEGER : age
 STRING  : str

DEBUT
    ECRIRE("Entrez votre age svp.")
    LIRE(age)

    str ⟵ CalculeAnneeDeNaissance(age)

    ECRIRE(str)
FIN
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

ynov
CAMPUS

EXERCICES

EXERCICE : ECHANGE DE VARIABLE

Ecrire un programme qui échange la valeur de deux variables.

EXERCICE : CALCUL DU CARRE

Ecrire un programme qui demande un nombre a l'utilisateur,
puis qui calcule et affiche le carré de ce nombre.

EXERCICE : LA CAISSE

Ecrire un programme qui lit le prix HT d'un article, le nombre d'articles et le taux de TVA, et qui fournit le prix total TTC correspondant.
Faire en sorte que des libellées apparaissent clairement

EXERCICE : LE PRODUIT

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif, positif ou nul.

V1 : Par le calcul
V2 : Sans calcul

LES STRUCTURES DE CONTRÔLE

Qu'est-ce qu'une structure de contrôle ?

LES STRUCTURES CONDITIONNELLES (IF)

```
ALGORITHME Exemple

DEBUT
        SI <condition> ALORS

                <instructions>

        FIN SI
FIN
```

LES STRUCTURES CONDITIONNELLES (IF ELSE)

```
ALGORITHME Exemple

DEBUT
        SI <condition> ALORS

                <instructions>

        SINON

                <instructions>

        FIN SI
FIN
```

LES STRUCTURES CONDITIONNELLES (IF ELSE IF ELSE)

```
ALGORITHME Exemple

DEBUT
        SI <condition> ALORS

                <instructions>

        SINON SI <condition> ALORS

                <instructions>

        SINON

                <instructions>

        FIN SI
FIN
```

LES CONDITIONS

LES OPÉRATEURS CONDITIONNELS

Condition ou test

```
Egal                    : =
Différents              : != ou <>
Inferieur à             : <
Supérieur à             : >
Inferieur ou égal à     : <=
Supérieur ou égal à     : >=

Opérateurs Booléens
```

STRUCTURE DE CHOIX (SWITCH)

```
ALGORITHME Exemple

DEBUT
        SELON <variable>
                <valeur1> :
                        <instructions>

                <valeur2> :
                        <instructions>

                AUTREMENT :
                        <instructions>
        FIN SELON
FIN
```

STRUCTURE DE RÉPÉTITIONS (WHILE)

```
ALGORITHME Exemple

DEBUT
        TANT QUE <condition> FAIRE

                <instructions>

        FIN TANT QUE
FIN
```

STRUCTURE DE RÉPÉTITIONS (DO WHILE)

```
ALGORITHME Exemple

DEBUT
        REPETER ( ou FAIRE )

                <instructions>

        TANT QUE <condition>
FIN
```

STRUCTURE DE RÉPÉTITIONS (FOR)

```
ALGORITHME Exemple

DEBUT
    POUR <variable> ALLANT DE <valeur début> A <valeur de fin> (PAR PAS DE <incrément>)

       <instructions>

    FIN POUR
FIN
```

LES STRUCTURES DE DONNÉES

LES STRUCTURES DE DONNÉES

LES STRUCTURES DE DONNÉES

Une structure de données est une manière d'organiser les données pour les traiter plus facilement.

Les structures finies
- Constantes,
- Variables,
- Enregistrement,
- Structure.

Les structures indexées
- Tableaux,
- Tableaux Multidimensionnelles,
- ~~Tableaux associatifs,~~
- Vecteurs.

Les structures récursives
- Listes,
- Arbres,
- Graphes.

LES STRUCTURES INDEXÉES

LES TABLEAUX

Un tableaux comporte :
- Un identifiant,
- Un type,
- Une taille

Chaque éléments est indexé.
On peut affecté une valeur a un indexe et récupérer une valeur par l'indexe.

Les indexes d'un tableau commence à 0.

```
ALGORITHME Exemple
VARIABLE

 TYPE : identifiant [taille]

DEBUT

 identifiant[2] ⟵ valeur

 AFFICHER( identifiant[2] )

FIN
```

LES TABLEAUX MULTIDIMENSIONNELLES

Un tableaux comporte :
- Un identifiant,
- Un type,
- Une taille
- Nombre de dimensions

Chaque éléments est indexé.
On peut affecté une valeur a un indexe et récupérer une valeur par l'indexe.

Les indexes d'un tableau commence à 0.

```
ALGORITHME Exemple
VARIABLE

 TYPE : identifiant [taille] [taille]
DEBUT

 identifiant[2][3] ⟵  valeur

 AFFICHER( identifiant[2][3] )

FIN
```

LES STRUCTURES RECURSIVES

LES STRUCTURES

Une structure est un type complexe pouvant être composé de plusieurs éléments de type différents.

```
STRUCTURE Point

    CHAINE Nom
    ENTIER x
    ENTIER Y

FIN STRUCTURE
```

```
ALGORITHME Exemple
VARIABLE

  Point : pts

DEBUT

  pts.Nom ←——  "BLABAL"
  pts.x   ←——  12
  pts.y   ←——  65

FIN
```

ASSOCIATIONS DE STRUCTURES

On peut associer les structures entre elles afin d'ordonner des éléments plus simple entre eux ou de pouvoir réaliser des structures récursives (piles, files listes etc.)

EXEMPLE : UN DICTIONNAIRE SIMPLE

```
STRUCTURE Mot

   String mot
   String definition

FIN STRUCTURE
```

```
STRUCTURE Dictionnaire

   Mot[10] tabMots

FIN STRUCTURE
```

Exemple 1 : un dictionnaire simple

```
STRUCTURE Mot

    String mot
    String definition

FIN STRUCTURE
```

```
STRUCTURE Dictionnaire

    Mot[10] tabMots

FIN STRUCTURE
```

EXERCICE 2 : ÉLÈVES ET NOTES

Représentez un élève sous forme de structure. Un élève est caractérisé par un son nom, son prénom, et des cours.

Un cours à un nom, et trois notes.

Ensuite implémenter un programmes, permettant au professeur de pouvoir enregistrer 5 élèves, et les trois notes par cours par élèves.

Le programmes affichera ensuite la liste des étudiants avec leur moyenne général et leur moyenne par cours.

Et enfin le programme affichera la moyenne général de la classe, et la moyenne général par cours.

CONCEPT DE LA LISTE

Une liste est une suite d'éléments de même nature.
Les éléments peuvent être ajouter, supprimé, trié etc.

LISTE CHAINÉE

LISTE CHAINÉE

Créer la liste chainée,
Implémenter les fonctions suivantes :
- Compter le nombre d'élèments de la liste
- Ajout d'un élément au début,
- Ajout d'un élément a la fin,
- Supprimer le premier élément
- Supprimer le deuxième élément
- Echanger l'élèment en deuxième position avec celui en quatrième position

LISTE DOUBLEMENT CHAINÉE

LISTE DOUBLEMENT CHAINÉE

Créer la liste chainée,
Implémenter les fonctions suivantes :
- Compter le nombre d'élèments de la liste
- Ajout d'un élément au début,
- Ajout d'un élément a la fin,
- Supprimer le premier élément
- Supprimer le deuxième élément
- Echanger l'élèment en deuxième position avec celui en quatrième position

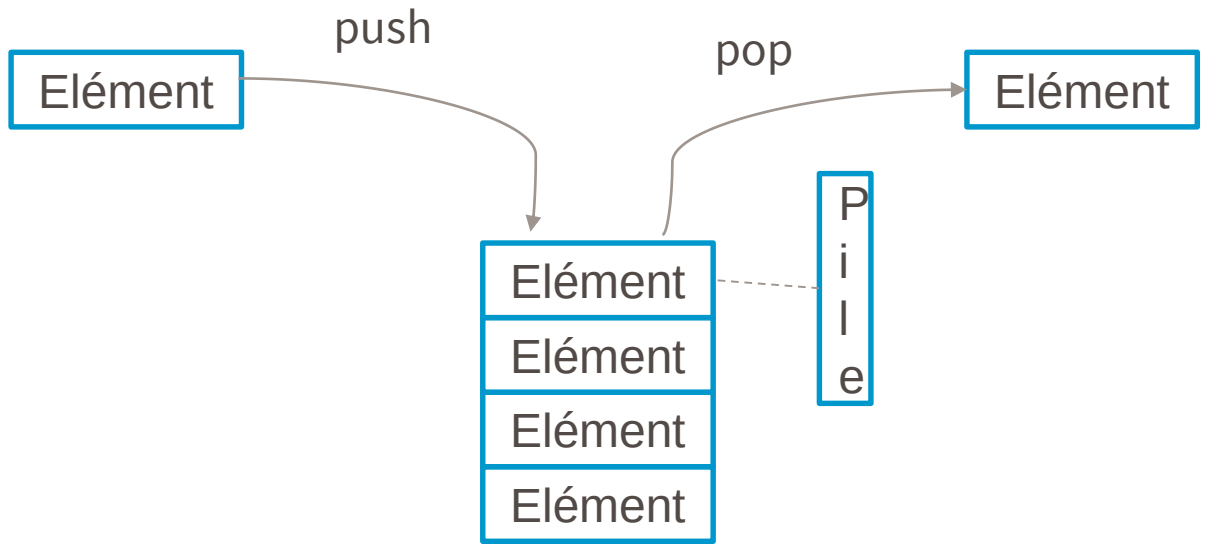
LISTE DOUBLEMENT CHAINÉE

```
// Remplacer le x-ème element avec le y-ème
// Exemple d'utilisatio :
//                      Swap_Elt(maListe, 3, 7)
//                      Swap_Elt(TaListe, 1000, 2)
//                      Swap_Elt(maaaaaaaaaListe, -3, 36)
// Liste des "cas limites" :
//    - Echanger un element avec lui meme
//              - Liste vide
//    - Un element au moins non présent
//    - x ou y négatifs
PROCEDURE Swap_Elt(Liste liste, ENTIER x ,
ENTIER y)
VARIABLE

DEBUT



FIN PROCEDURE
```

CONCEPT DE LA PILE

C'est une liste avec la propriété : LIFO (last in, first out)

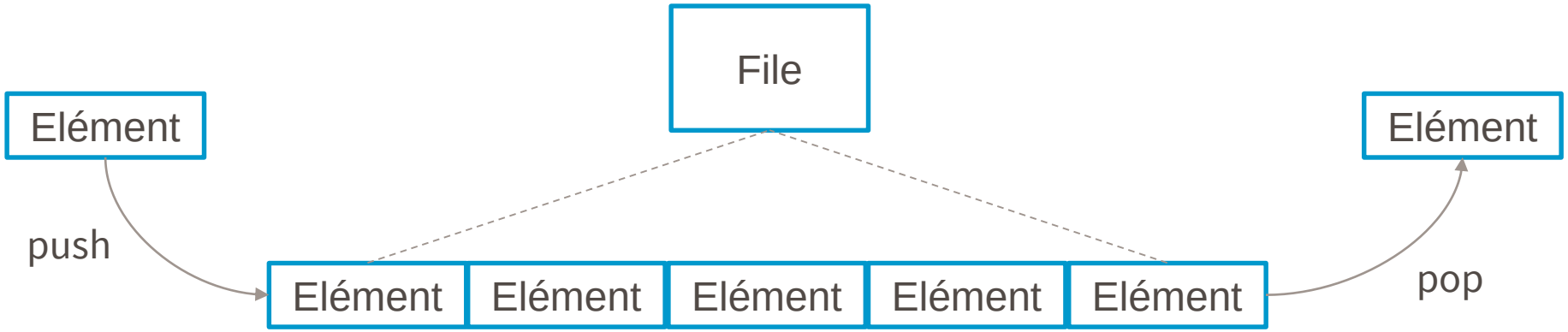
Créez la ou les structures définissant la pile
- Créer les fonctions Push
- Créer les fonction Pop
- Compter éléments d'une pile
- Supprimer le dernier élément de la pile
- Ajouter un élément à la fin de la pile
- Enlever un X-ème élément de la pile
- Ajouter un élément a la positio X



CONCEPT DE LA FILE

C'est une liste avec la propriété : FIFO (first in, first out)

Créez la ou les structures définissant la file
- Créer les fonctions Push
- Créer les fonction Pop
- Compter éléments d'une file
- Vider la file complétement



LES ALGORITHMES DE TRI

LES ALGORITHMES DE TRI

LES ALGORITHMES DE TRI

- ✓ **Tri par sélection**
- ✓ **Tri par insertion**
- ✓ **Tri à bulle**
- ✓ **Tri fusion**
- ✓ **Tri rapide**
- ✓ **Tri avec les arbres**
- ✓ **Etc.**

LE TRI PAR SÉLECTION

- Trouver le plus petit élément et le mettre en première position
- Trouver le second plus petit élément et le mettre en seconde position
- Trouver le troisième plus petit élément et le mettre en troisième position
- …
- Trouver le N-ème plus petit élément et le mettre en N-ème position.

5 81 6 2 89 1 5 => 1 2 5 6 81 89

```
// Le tableau retourné doit être trié par selection
FONCTION TriParSelection (Entier tab[], Entier taille ) : Entier []
VARIABLE

DEBUT

FIN FONCTION
```

LE TRI PAR INSERTION

Insérer le nouvel élément de manière à ce que la liste soit trié.

LE TRI À BULLE

6	0	3	5	1	4	2
0	6	3	5	1	4	2
0	3	6	5	1	4	2
0	3	5	6	1	4	2
0	3	5	1	6	4	2
0	3	5	1	4	6	2
0	**3**	**5**	**1**	**4**	**2**	**6**

LE TRI À BULLE

0	3	5	1	4	2	6
0	3	5	1	4	2	6
0	3	5	1	4	2	6
0	3	1	5	4	2	6
0	3	1	4	5	2	6
0	3	1	4	2	5	6
0	**3**	**1**	**4**	**2**	**5**	**6**

LE TRI À BULLE

5 81 6 2 89 1 5 => 1 2 5 6 81 89

// Le tableau retourné doit être trié par selection

FONCTION TriABulle (Entier tab[], Entier taille)
VARIABLE

DEBUT

FIN FONCTION

LE TRI FUSION

Permet de fusionner deux listes de données triés pour former une seul et unique liste trié.

LE TRI FUSION

3 5 7 10 12 13 15 16 19 20 25 35

```
FONCTION TriFusion (Entier tab1[], Entier tab2[], Entier taille ) : Entier[]
VARIABLE
DEBUT

FIN FONCTION
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.

A propos de Capgemini
Avec plus de 190 000 collaborateurs, Capgemini est présent dans plus de 40 pays et célèbre son cinquantième anniversaire en 2017. Le Groupe est l'un des leaders mondiaux du conseil, des services informatiques et de l'infogérance et a réalisé en 2016 un chiffre d'affaires de 12,5 milliards d'euros. Avec ses clients, Capgemini conçoit et met en œuvre les solutions business, technologiques et digitales qui correspondent à leurs besoins et leur apportent innovation et compétitivité. Profondément multiculturel, Capgemini revendique un style de travail qui lui est propre, la « Collaborative Business ExperienceTM », et s'appuie sur un mode de production mondialisé, le « Rightshore® ».

www.capgemini.com