

Principes de programmation (PP)

0. INTRODUCTION

Plan du cours

- 0. Introduction
- 1. Variables
- 2. Alternatives & Conditions
- 3. Répétitives
- 4. Tableaux

Informations

- UE de programmation
 - AA : Langage
 - AA : Principes
- **Théorie** : 8 h -> 4 séances : auditoire
- **Exercices** : 16 h -> 8 séances : par groupe
- **Examen** en janvier (et éventuellement en juin, et éventuellement en septembre) : réussite à partir de 10/20
- Evaluation intégrée !

Qu'est ce que la programmation ?

Programmation :

Ensemble des activités qui permettent l'écriture de programmes informatiques

Programme :

Ensemble d'instructions (écrites dans un langage de programmation) destinées à être exécutées par un ordinateur

Qu'est-ce qu'un algorithme ?

Suite d'instructions simples qui permet d'obtenir un résultat.

- algorithme **exact** \Rightarrow le résultat est celui **attendu**
- algorithme **inexact** \Rightarrow le résultat est **indéfini**...

Exemples

- Recette de cuisine
- Notice de montage
- Résolution du Rubik's Cube
- Calcul du $n^{\text{ième}}$ nombre de la suite de Fibonacci

\rightarrow Ce cours porte **sur la conception d'algorithmes**

Principes de programmation

Méthodes pour une construction réfléchie, rigoureuse et efficace d'**algorithmes**.

→ Écrire les algorithmes indépendamment des particularités de tel ou tel langage de programmation structurée.

Écriture ?

- organigrammes
- GNS
- pseudo-code
- **diagrammes d'actions (DA)**

Diagrammes d'actions

Comment ?

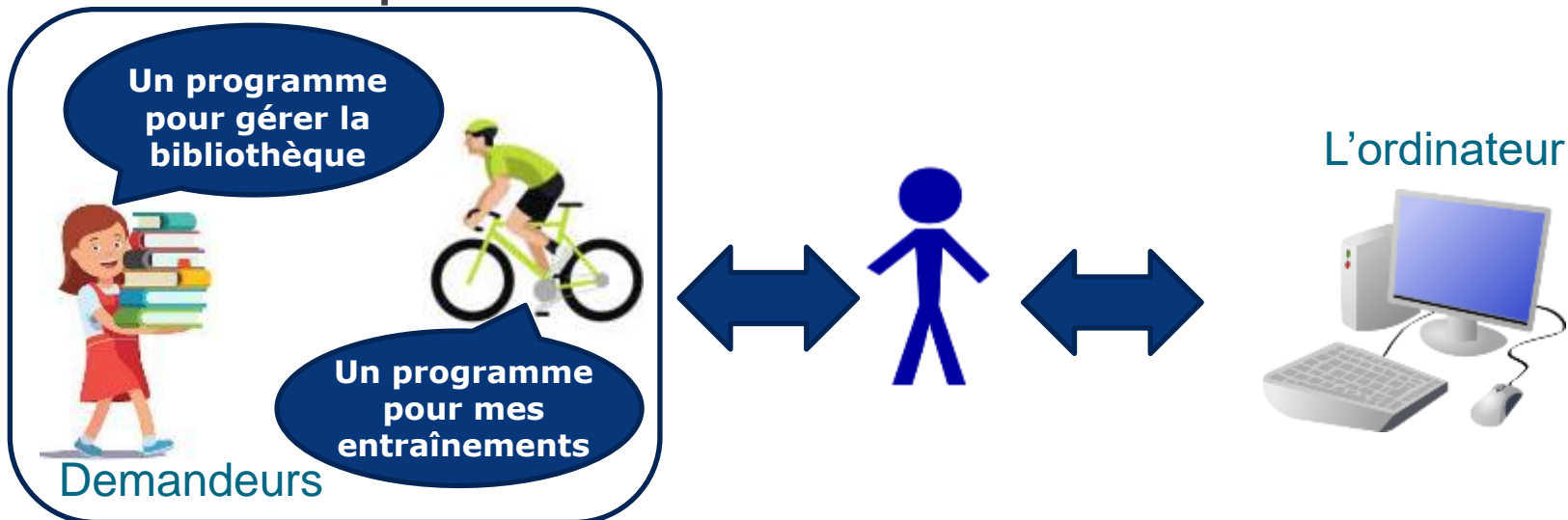
- sur papier ou ...
- Sur PC : grâce à Gaëtan Le Docte
 - En ligne : <https://section-ig.github.io/da/>

Analyser avant de programmer

programmer = concevoir et écrire des programmes

1^{ère} étape : Analyser

structurer les désirs du demandeur en « quelque chose d'implémentable ».



Analyser avant de programmer

Première étape pour structurer un problème :

distinguer les

entrées

Les données que le programme va recevoir/utiliser, dont il a besoin.

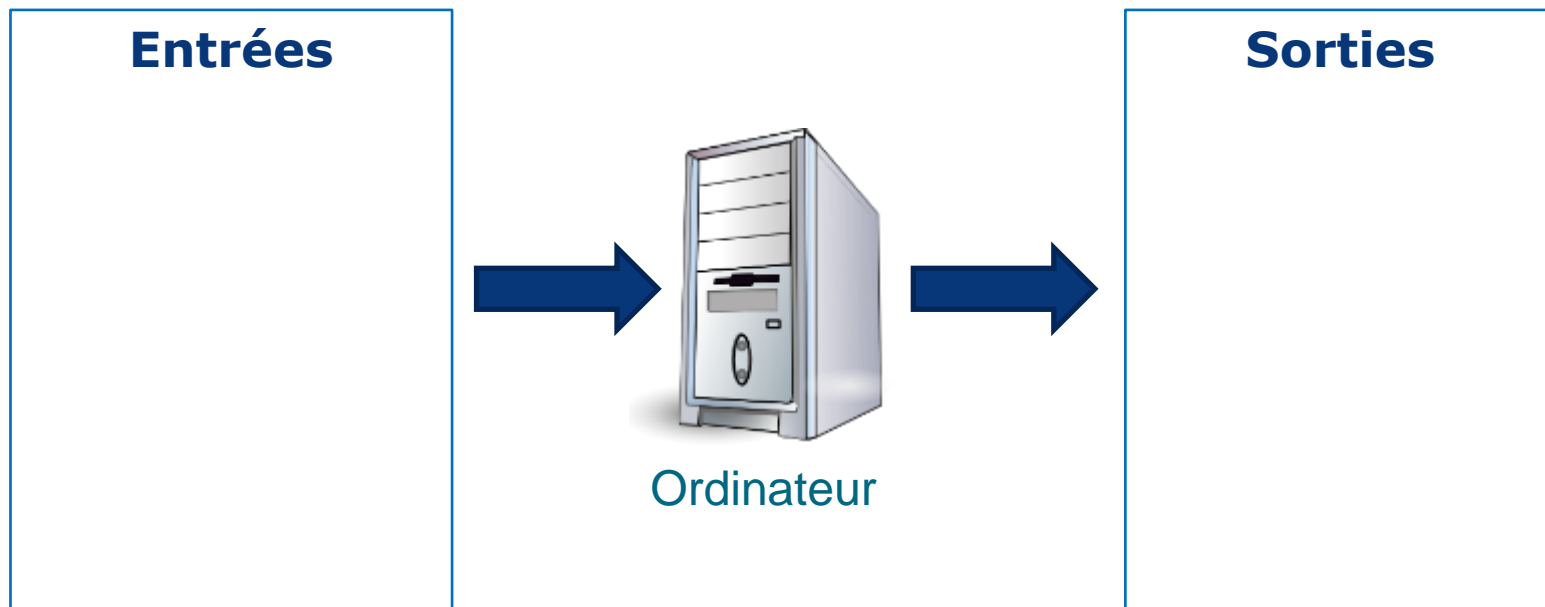
des

sorties

Les résultats que le programme va produire

Principes de programmation

- Il existe de nombreux périphériques d'entrée / de sortie :



Exemple 1 : le cycliste

Un cycliste amateur se prépare énergiquement pour une course qui est organisée à la fin du mois. Chaque jour, il prend son vélo et effectue un parcours soigneusement choisi pour correspondre au plus près à celui de la véritable course. Il lance son chronomètre avant de partir et, dès son retour, note le temps qu'il a effectué dans son calepin. Écris un programme qui lui permettra d'entrer les temps de ses différents entraînements de la semaine (du lundi au dimanche) au format **HHMMSS** et lui indiquera son pire temps et son meilleur temps.

*Entrées

*Sorties



Format HHMMSS

Quels sont les avantages (informatiques) à utiliser le format HHMMSS ?

Format HHMMSS

Complétez le tableau suivant :

Format usuel	Format HHMMSS
10 heures, 11 minutes, 12 secondes	
5 heures, 6 minutes, 7 secondes	
3 heures et 4 secondes	
	170519
	140000
	30125

Pourquoi prendre la peine d'ajouter des zéros aux nombres inférieurs à 10 ?

Le cycliste : entrées

Voici la page du calepin du cycliste correspondant à ses entraînements de la semaine. Indiquez précisément ce qu'il devra entrer pour utiliser le programme.

Lundi - 1h 19min
Mardi : 58' 30"
Mer : 1h tout pile !
Jeu : 81 min 15 sec
Ven : 1h 3' 40"
Sam - 1 h et demi (j'aurais pas dû sortir vendredi)
Dim : 62min 10sec

**De nombreux
formats
différents**

Le cycliste : sorties

Si le cycliste effectue les entrées correctement (voir question précédente) et que le programme est correct, quels seront les résultats affichés ?

Lundi - 1h 19min	011900
Mardi : 58' 30"	005830
Mer : 1h tout pile !	010000
Jeu : 81 min 15 sec	012115
Ven : 1h 3' 40"	010340
Sam - 1 h et demi	013000
(j'aurais pas dû sortir vendredi)	010210
Dim : 62min 10sec	

Format AAAAMMJJ

Complétez le tableau suivant.

Format usuel	Format AAAAMMJJ
	20150901
	20150717
1 ^{er} mars 2017	
15 janvier 2000	

Exercice 2 : la bibliothèque

La responsable d'une bibliothèque désire encoder les informations reprises dans les fiches concernant les membres inscrits dans son établissement. Chaque membre est caractérisé par une date d'inscription, un nom, une date de naissance et une localité. À la fin de chaque semaine, elle compte mettre à jour le fichier en ajoutant les informations relatives aux nouveaux inscrits. Comme la bibliothèque est sponsorisée par la ville de Namur, elle aimerait connaître chaque semaine le nombre de nouveaux inscrits habitant à Namur. De plus, afin de mieux adapter le choix de livres proposé, elle voudrait connaître chaque semaine le nombre de nouveaux inscrits qui ont moins de 20 ans.

Note. Les dates seront encodées au format AAAAMMJJ.

*Entrées

*Sorties



La bibliothèque : sorties

Le tableau suivant reprend les informations entrées par la bibliothécaire le 29 septembre 2025. Quels résultats le programme devra-t-il afficher ?

Date d'inscription	Nom	Date de naissance	Localité
20250924	A. Squelle	20100507	Charleroi
20250924	B. Zique	19970108	Namur
20250925	C. Charpe	20050212	Namur
20250925	F. Charpe	20051219	Namur
20250926	K. Melle	20020713	Huy
20250927	P. Hachepée	20100405	Dinant
20250929	S. Kiuelle	19821125	Namur

1. VARIABLES

Welcome

Imaginons un petit programme « Welcome »...

```
Comment t'appelles-tu ? Maxime  
Bonjour Maxime.  
En quelle année es-tu né ? 2001  
Oh, 20 ans cette année !  
Au revoir Maxime.
```

- Nécessité de mémoriser "Maxime" pour une réutilisation ultérieure
- Nécessité de mémoriser "2001" pour effectuer le calcul de l'âge

Besoin de stocker

Régulièrement besoin de stocker des valeurs : valeurs introduites par l'utilisateur, valeurs intermédiaires de calcul, ...

→ Une variable va **stocker une information**.
À un moment donné, elle contiendra une et une seule valeur.

Elle est caractérisée par :

- un **type**
- un **nom**
- un **contenu**

Le type

3 types possibles :

- **Numérique** : nombre naturel, entier, réel
→ opérations arithmétiques autorisées
- **Chaine de caractères** : caractères alphabétiques, numériques ou autres
- **Booléen** : 2 valeurs logiques, VRAI ou FAUX, TRUE ou FALSE

Une fois le type de variable choisi, il **ne peut pas** être changé !

Le contenu

Littéral : représentation en toutes lettres du contenu d'une variable. Sa forme dépend du type :

- **Numérique** : 42 12,8 -12587
- **Chaine de caractères** : "placé entre guillemets"
- **Booléen** : true ou false

Le contenu d'une variable **peut** être modifié lors de l'exécution du programme.

Affectation

L'affectation (=) consiste à attribuer une valeur (à droite) à une variable (à gauche)



- nombreEtudiantsIG = 211
- nomClient = "Jacques Dulieu"
- valeurMax = valeur



- prixTotal = prixHTVA + TVA

Le nom


Rôle mnémotechnique

- En un seul mot (pas d'espace possible)
- Convention : débute par une minuscule, majuscule au début de chaque mot (lower camel case)

→ Exemples : `nombreEtudiantsIG,`
`nomClient, estMultiple97,`
`fraisPortOfferts, ...`

Une fois défini, le nom d'une variable **ne peut pas** changer en cours de programme

Affectation

- `somme = somme + 10`
- `prixTotal = prixTotal + prix`
- `estMultiple97 = nombre % 97 == 0`
- `fraisPortOfferts = montantAchats > 60`

Premier DA : la foire

Une famille se promène à la foire du midi et décide d'acheter des beignets. En fonction du nombre de beignets achetés et du prix d'un beignet, écrivez le DA permettant de calculer et d'afficher le montant à payer.

Commencez par déterminer les entrées et les sorties.

Opérateurs arithmétiques : $+$, $-$, $*$, $/$, $\%$

2. ALTERNATIVES & CONDITIONS

Exemple : la foire

Une famille se promène à la foire du midi et décide d'acheter des beignets. En fonction du nombre de beignets achetés et du prix d'un beignet, écrivez le DA permettant de calculer et d'afficher le montant à payer sachant qu'une réduction de 5% est accordée pour tout achat de 6 beignets ou plus.

Les alternatives

```
if (expression conditionnelle)  
// instructions à exécuter si l'expression conditionnelle est vraie
```

```
if (expression conditionnelle)  
// instructions à exécuter si l'expression conditionnelle est vraie  
else  
// instructions à exécuter si l'expression conditionnelle est fausse
```

Opérateurs : == ≠ < > ≤ ≥

Exemple : la foire

Une famille se promène à la foire du midi et décide d'acheter des beignets. En fonction du nombre de beignets achetés et du prix d'un beignet, écrivez le DA permettant d'afficher la réduction et le montant à payer sachant qu'une réduction de 5% est accordée pour tout achat de 6 beignets ou plus.

Exemple : la foire

Une famille se promène à la foire du midi et décide d'acheter des beignets. En fonction du nombre de beignets achetés et du prix d'un beignet, écrivez le DA permettant de calculer et d'afficher le montant à payer.

Une réduction de 5% est accordée pour tout achat de 6 beignets ou plus. La réduction est doublée si le client achète plus de 15 beignets.

Exemple : grande surface

Une grande surface accorde une réduction de prix pour des achats multiples.

- 10% sur le prix à l'unité à partir de 3 unités achetées
- 15% sur le prix à l'unité à partir de 5 unités achetées
- 25% sur le prix à l'unité à partir de 10 unités achetées

Sur base du **prix unitaire** et de la **quantité** de produit acheté, calculez et affichez le **prix à payer**.

Exemple : sirop

La notice d'un sirop est libellée comme suit :

- Adultes de plus de 50 kg, 90 ml par jour maximum
- Enfants de 30 à 50 kg, 60 ml par jour maximum
- Enfants de 15 à 30 kg inclus, 30 ml par jour maximum
- Ne pas administrer aux enfants de moins de 15 kg

Déterminer la dose maximale à administrer à une personne dont on obtient le poids en kg.

Exemple : livraison

À partir du prix unitaire et de la quantité commandée d'un produit, écrire le DA qui calcule et affiche la remise et les frais de port ainsi que le prix total à payer.

Le port est gratuit si le total à payer est supérieur à 25€, sinon il vaut 2% du total.

La remise est de 5% si le total à payer est compris entre 10€ et 50€ et de 10% au-delà.

MODULES

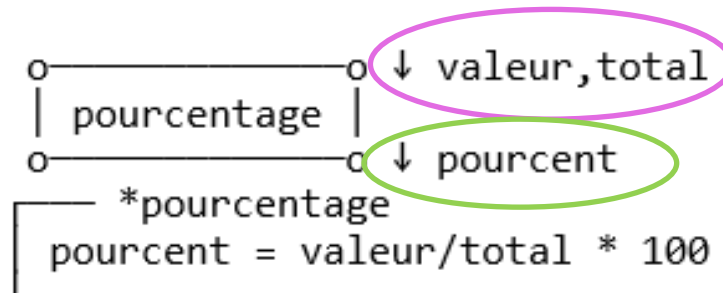
Découpe en modules

Un module est un **algorithme** avec une **fonction précise** (calculer une réduction, obtenir une date valide, valider une date, ...)

- Quand créer un module ?
 - Lorsqu'une suite d'instructions a un **rôle précis**
 - Chaque fois qu'une même suite d'instructions doit être **répétée** dans un DA
- Pourquoi ?
 - Rendre le DA plus **lisible/clair**
 - **Faciliter** les modifications/mises à jour
→ **Point de modification unique**

Les paramètres

Le module peut recevoir des données (**paramètres d'entrée**) et/ou renvoyer un résultat (**paramètres de sortie**, retour)

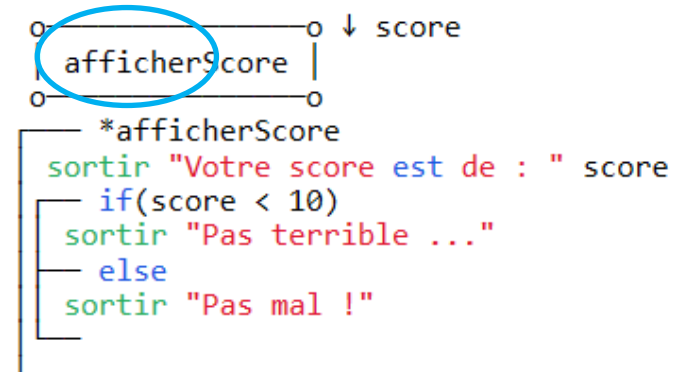
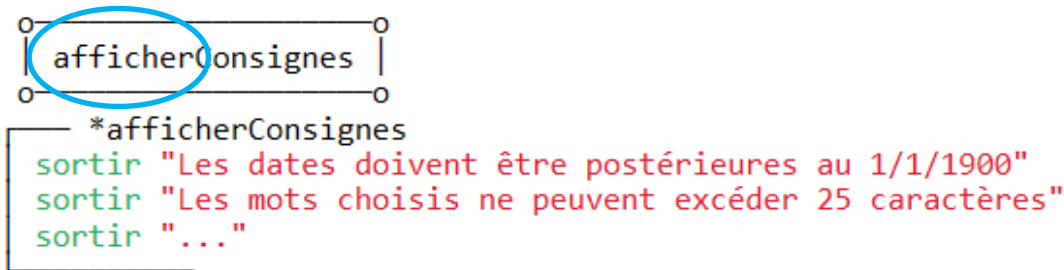


**/!\ paramètres d'entrée et sortie
≠
« obtenir » et « sortir »**

Choix du nom

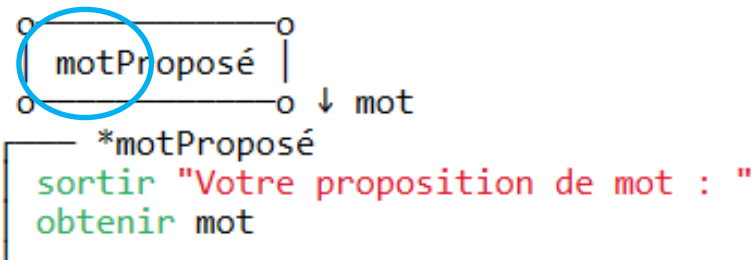
Pas de paramètre de sortie = correspond à une **action**

→ le nom du module contient un **verbe**



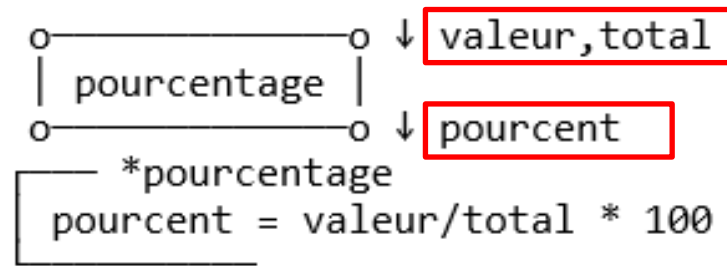
Paramètre de sortie = correspond à une **valeur**

→ le nom du module contient un **nom** qui correspond à ce qu'il est chargé de fournir au programme

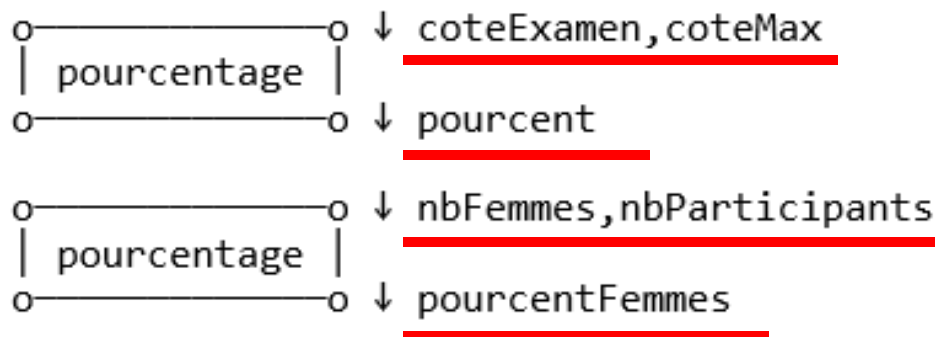


Appels

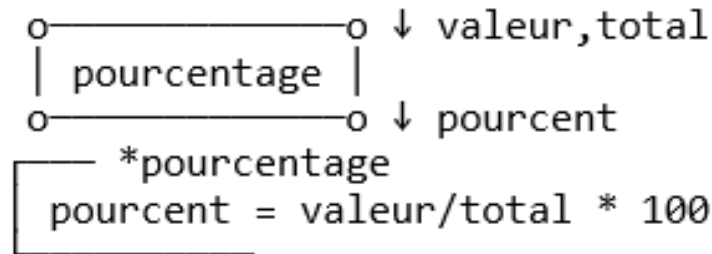
- Dans la **description** du module, on reste le plus **générique** possible.



- Lors des différents **appels**, on **précise** les entrées et sorties .



pré et post conditions



Division par 0 possible ? Et si valeur est < 0 ? ...

→ Pré-condition(s) = **condition(s) à vérifier** afin que le module fasse bien ce qui est prévu.

Ici : total > 0 et valeur ≥ 0

→ Post-condition(s) = ce que le module **s'engage à réaliser** si les pré-conditions sont vérifiées.

Ici : pourcent contiendra le rapport en % entre valeur et total.

pré et post conditions : exemples

```
o-----o ↓ entier, diviseur  
| afficheSiDiviseur |  
o-----o
```

```
// pré-condition : diviseur est un nombre entier > 0  
// post-condition : affiche un message indiquant si  
                    diviseur est diviseur de entier
```

```
    *  
    [ if entier % diviseur == 0  
      [ sortir diviseur, " est diviseur de ", entier  
    ]  
    [ sortir diviseur, " n'est pas diviseur de ", entier  
  ]
```

3. RÉPÉTITIVES

Température de Namur

Sachant que l'on obtient la température à midi à Namur de chaque jour du mois de septembre, écrire le DA qui affiche la température moyenne à midi en septembre à Namur.

Entrées :

température1, température2, ..., température30

Sortie :

températureMoyenne

Température en septembre

Version 1

```
0-----0  
| établirTempératureMoyenneSeptembre |  
0-----0
```

*

```
obtenir température1, température2, ..., température30
```

```
somme = température1 + température2 + ... + température30
```

```
températureMoyenne = somme/30
```

```
sortir températureMoyenne
```

Température en septembre

Version 3

```
0-----0  
| établirTempératureMoyenneSeptembre |  
0-----0
```

*

```
somme = 0
```

```
faire 30 fois
```

```
obtenir température
```

```
somme += température
```

```
températureMoyenne = somme/30
```

```
sortir températureMoyenne
```

Entrées :

température
(30 *)

À tout moment, *somme*
contiendra le total des
températures obtenues

Température en septembre

Version 4 optimale

```
0-----0  
| établirTempératureMoyenneSeptembre |  
0-----0
```

```
*  
somme = 0  
  
nbTempératures = 0  
while (nbTempératures < 30)  
    obtenir température  
    somme += température  
    nbTempératures ++  
  
températureMoyenne = somme/nbTempératures  
sortir températureMoyenne
```

On continue tant qu'on a pas 30 t°
= Tant que nbTempératures < 30

nbTempératures compte le nombre de passages dans la boucle
(= le nombre de t° obtenues)

- 0 au départ
- +1 à chaque passage

Température en septembre

Version 2

```
0-----0  
| établirTempératureMoyenneSeptembre |  
0-----0
```

```
  *  
  somme = 0  
  
  obtenir température1 } 1 fois  
  somme += température1  
  
  obtenir température2 } 2 fois  
  somme += température2  
  ...  
  obtenir température30 } 30 fois  
  somme += température30  
  
  températureMoyenne = somme/30  
  sortir températureMoyenne
```

Entrées :
température
(30 *)

Température moyenne

On ne connaît pas le nombre de températures.

- 1^e possibilité : demander à l'utilisateur le nombre de températures qu'il compte introduire.

```
0-----0
| établirTempératureMoyenne |
0-----0

*
somme = 0
sortir "Nombre de températures ?"
obtenir nbTempératures
cptTempératures = 0
while (cptTempératures < nbTempératures)
  obtenir température
  somme += température
  cptTempératures ++
if (nbTempératures > 0)
  températureMoyenne = somme/nbTempératures
else
  températureMoyenne = 0
sortir températureMoyenne
```

Critiques ?

- ✓ L'utilisateur doit compter
- ✓ Il peut se tromper

Température moyenne

Sachant que l'on obtient la température à midi à Namur de chaque jour **d'une période quelconque**, écrire le DA qui affiche la température moyenne à midi à Namur sur cette période.

Entrées :
température
(t *)

t inconnu

Température moyenne

- 2^e possibilité : demander à l'utilisateur, à chaque fois, s'il souhaite continuer.

```
0-----0
| établirTempératureMoyenne |
0-----0

*
somme = 0
réponse = "oui"
nbTempératures = 0
while (réponse == "oui")
    obtenir température
    somme += température
    nbTempératures ++
    sortir "Encore? (oui/non)"
    obtenir réponse

if (nbTempératures > 0)
    températureMoyenne = somme/nbTempératures
else
    températureMoyenne = 0

sortir températureMoyenne
```

Critiques ?

- ✓ Pas convivial
- ✓ Trop lourd pour l'utilisateur

Température moyenne

- 3^e possibilité : convenir d'une valeur « bidon » pour quitter la boucle.

```
0-----0
| établirTempératureMoyenne |
0-----0

*
somme = 0
nbTempératures = 0
températures = 0
while (température ≠ -100)
  sortir "température ? (-100 pour terminer)"
  obtenir température
  if température ≠ -100
    somme += température
    nbTempératures ++
  if (nbTempératures > 0)
    températureMoyenne = somme/nbTempératures
  else
    températureMoyenne = 0
sortir températureMoyenne
```

Critiques ?

- ✓ Test d'arrêt présent 2 fois
- Pas efficace
- Pas optimal

Température moyenne

- 3^e possibilité : convenir d'une valeur « bidon » pour quitter la boucle.

```
0-----0
| établirTempératureMoyenne |
0-----0

*
somme = 0
nbTempératures = 0
températures = 0
obtenir température
while (température ≠ -100)
    somme += température
    nbTempératures ++
    obtenir température
if (nbTempératures > 0)
    températureMoyenne = somme/nbTempératures
else
    températureMoyenne = 0
sortir températureMoyenne
```

Version optimale

Résumé

```
init  
[ while ( cond )  
  ...  
  màj
```

Si **on connaît** le nombre d'itérations: Si **on connaît** le nombre d'itérations:

```
cpt = 0  
[ while (cpt < nb)  
  ...  
  cpt ++
```

```
obtenir info  
[ while (info ≠ -1)  
  ...  
  obtenir info
```

Touché-coulé

Chacun des 2 joueurs désigne, à tour de rôle, une case (numéro de ligne de 1 à 10 et numéro de colonne de A à J). Écrire le DA qui obtient un « coup » et ce jusqu'à ce qu'il corresponde bien à une case du damier. Afficher ensuite les coordonnées obtenues.

Entrées :

pour chaque essai $\begin{cases} \text{numLigne} \\ \text{numColonne} \end{cases}$
(e *)

Sorties :

numLigne
numColonne

Touché-coulé

- On ne connaît pas le nombre d'essais au départ.

```
o-----o
| coup |
o-----o ↓ ligne, colonne

*
  obtenir ligne, colonne
  while (ligne<1 OR ligne>10 OR colonne<"A" OR colonne>"J" )
    obtenir ligne, colonne
```

Taille

Écrire un DA qui permet d'établir et d'afficher la taille du plus grand étudiant d'une classe. On introduira les tailles en cm de chaque étudiant (0 pour terminer).

Entrées :
pour chaque étudiant { taille
(e *)

Sorties :
tailleMax

Taille

Écrire un DA qui permet d'établir et de sortir la taille moyenne des étudiants de la classe, ainsi que la taille minimum et la taille maximum. On introduira les tailles en cm de chaque étudiant (0 pour terminer).

Entrées :

pour chaque étudiant { taille
(e *)

Sorties :

tailleMoyenne
tailleMax
tailleMin

Concours de saut

Lors d'une fête caritative, un concours de saut en longueur est organisé. Pour chaque participant, on obtient son nom, son âge, un code sexe (H/F) et pour chacun des 5 sauts effectués, la longueur du saut en cm.

Écrire le DA qui affiche :

1. par participant : son nom, la longueur moyenne des sauts, la longueur du saut le plus long
2. le nom du participant qui a effectué le meilleur saut
3. le nom du plus jeune participant et son meilleur saut
4. la répartition en pourcents entre les sexes

Concours de saut

Entrées :

pour chaque participant
(p *)

$$\left\{ \begin{array}{l} \text{nom} \\ \text{age} \\ \text{codeSexe} \\ \text{saut} \\ (5 \ *) \end{array} \right.$$

Sorties :

pour chaque participant
(p *)

$$\left\{ \begin{array}{l} \text{nom} \\ \text{longueurMoyenne} \\ \text{longueurMax} \end{array} \right.$$

nomLongueurMax

nomCadet, longueurMaxCadet

pourcHommes, pourcFemmes

EXERCICES RÉCAPITULATIFS

Démarche

1. Structures E/S pour bien comprendre l'énoncé
2. Répétitive principale (laquelle ?) + placer les obtentions
3. Placer les sorties (dedans ou dehors?)
4. Décomposer le reste en fonction des sorties à produire, en séparant les traitements.

Distance domicile-école

Sachant que, pour chaque étudiant, on obtient son nom et la distance qu'il doit parcourir de son domicile à l'école, écrire le DA qui calcule et affiche la distance minimum (et le nom de l'étudiant concerné), la distance maximum (et le nom de l'étudiant concerné) et la répartition en pourcents des étudiants en fonction de la distance domicile/école : moins de 1 km, de 1 à moins de 5 kms, de 5 à moins de 10 kms et enfin, 10 kms et plus.

(Les distances sont exprimées en kms et le dernier nom obtenu sera zzz)

Diviseurs et factorielle

Étant donné une série de nombres entiers strictement positifs (à valider, dernier nombre obtenu : 0), écrivez le DA permettant de calculer et d'afficher, pour chacun, la liste de ses diviseurs et sa factorielle.

4. TABLEAUX

Concours de déguisement

Lors d'un concours de déguisement, 10 personnes costumées se présentent face au public. Chaque membre du public est invité à voter en introduisant le numéro du déguisement qu'il préfère (de 1 à 10). On demande d'écrire le DA qui obtient chaque vote et affiche ensuite, pour chaque déguisement, son numéro et le nombre total de votes reçus.

Entrées :

vote

(v *)

Sorties :

1, nbVotes1

2, nbVotes2

...

10, nbVotes10

Concours de déguisement

```
graph TD
    subgraph comptervotes [compterVotes]
        direction TB
        nbVotes1[0]
        nbVotes2[0]
        nbVotes10[0]
        obtenirVote[obtenir vote]
        doWhile[do while (vote != 0)]
        doWhile --> if1[if (vote == 1)]
        doWhile --> if2[if (vote == 2)]
        doWhile --> if3[if (vote == 3)]
        doWhile --> etc[... etc ...]
        obtenirVote --> doWhile
    end
    comptervotes --> sortie[sortir "1", nbVotes1  
sortir "2", nbVotes2  
...  
sortir "10", nbVotes10]
```

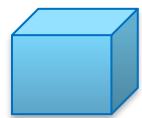
```
*compterVotes
nbVotes1 = 0
nbVotes2 = 0
...
nbVotes10 = 0
obtenir vote
do while (vote ≠ 0)
    if (vote == 1)
        nbVotes1 ++
    if (vote == 2)
        nbVotes2 ++
    if (vote == 3)
        ... etc ...
obtenir vote
sortir "1", nbVotes1
sortir "2", nbVotes2
...
sortir "10", nbVotes10
```

Critiques ?

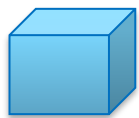
- ✓ Laborieux
- ✓ Et si on a 50 candidats ou plus ?
- ✓ Et si on ne connaît pas le nombre de candidats au préalable ?

Peut mieux faire ?

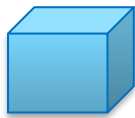
- Chacune des variables a le même rôle de compter les votes.



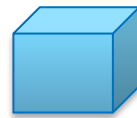
nbVotes1



nbVotes2



nbVotes3



nbVotes4

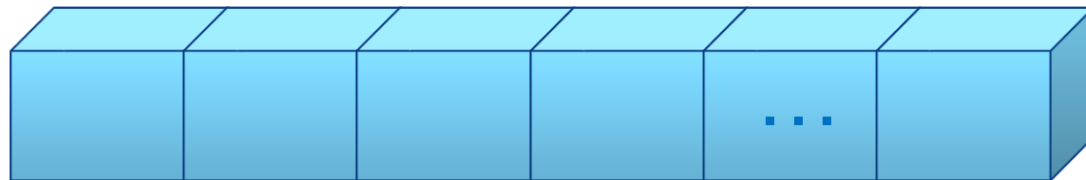
...



nbVotes10

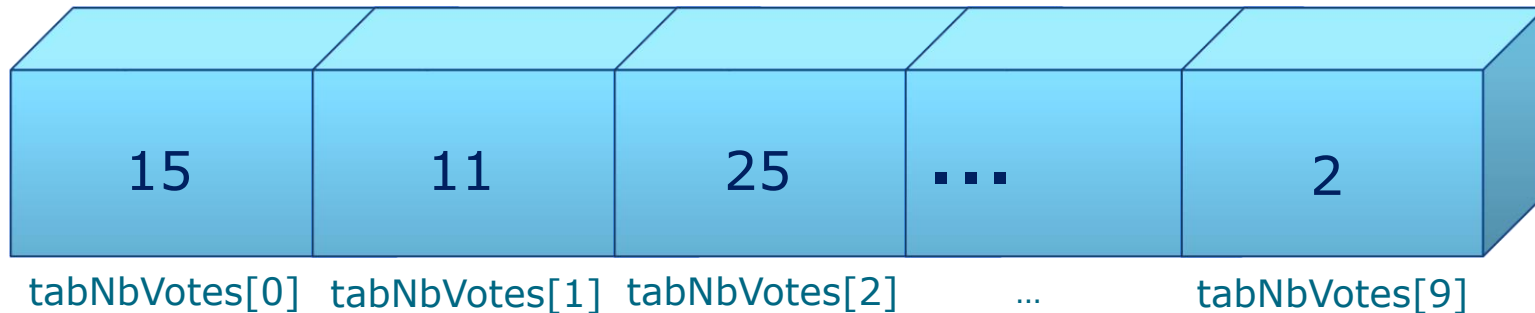
- On va regrouper les différentes valeurs de comptage dans une seule variable : **un tableau.**

tabNbVotes



- Structure : tabNbVotes $\left\{ \begin{array}{l} \text{cellule} \\ (10 *) \end{array} \right.$

Accès



- `tabNbVotes` est le nom du tableau
- Pour accéder au contenu d'une cellule :
`tabNbVotes[indice]`

Remarque : l'indice d'un tableau commence à 0. Un tableau comprenant N éléments verra son indice aller de 0 à $N-1$.

Définition

Le tableau est :

- une **collection** de données
- **homogènes**
- **à accès direct** :
- où les éléments sont **consécutifs en mémoire**
- et dont l'allocation mémoire est **statique** (donc de **taille fixe**)

Définition

Le tableau est :

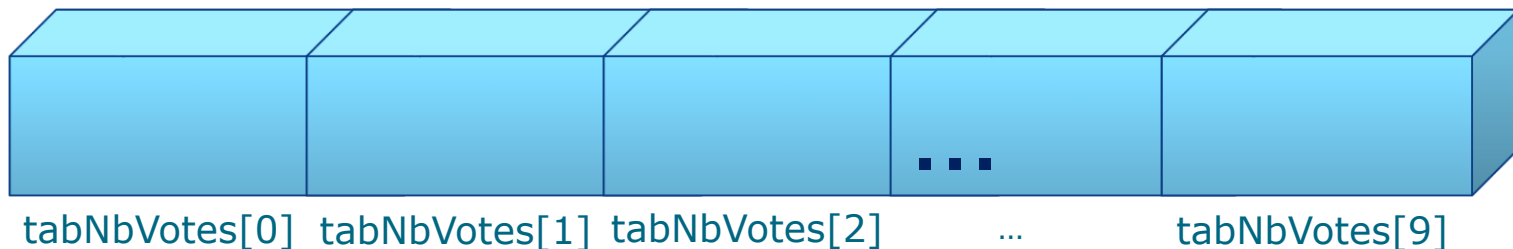
- une **collection** de données : regroupe plusieurs éléments
- **homogènes** : chaque élément est du même type
- **à accès direct** : il suffit de connaître l'indice pour accéder à un élément
- où les éléments sont **consécutifs en mémoire**
- et dont l'allocation mémoire est **statique** : une fois l'espace mémoire alloué, il ne peut plus changer
→ **taille fixe** : ne peut pas changer

Déclaration

Un tableau est une structure de données statique, c'est-à-dire qu'il est impossible de supprimer ou d'ajouter une cellule après avoir **déclaré le tableau**.

Pour déclarer un tableau, nous utiliserons le mot **ARRAY** suivi du nombre de case à prévoir entre parenthèses

→ `tabNbVotes = ARRAY(10)`



Concours de déguisement

Lors d'un concours de déguisement, 10 personnes costumées se présentent face au public. Chaque membre du public est invité à voter en introduisant le numéro du déguisement qu'il préfère (de 1 à 10). On demande d'écrire le DA qui obtient chaque vote et affiche ensuite, pour chaque déguisement, son numéro et le nombre total de votes reçus.

Entrées :

vote
(v *)

Sorties :

pour chaque déguisement $\begin{cases} \text{numéro} \\ \text{nbVotes} \end{cases}$
(10 *)

À créer :

tabNbVotes $\begin{cases} \text{cellule} \\ (10 *) \end{cases}$

Concours de déguisement

```
○-----○  
| AfficherNbVotes |  
○-----○  
  
┌ *  
│ ...  
│  
│ obtenir vote  
│ ┌ while (vote ≠ 0)  
│ │ ...  
│ │ obtenir vote  
│ └─┘  
│ ...  
└─┘
```

Initialiser le tableau avec 10
cellules à 0

Ajouter 1 dans la bonne cellule

Sortir les résultats

Les élections

7 partis se présentent aux élections et reçoivent chacun un numéro de 1 à 7. Après un sondage auprès d'un certain nombre d'électeurs sur leurs intentions de vote, on souhaite connaître le pourcentage estimé de voix pour chaque parti ainsi que le pourcentage d'indécis.

Écrivez le DA correspondant.

Un électeur indécis entrera un choix de 8.

Les élections

7 partis se présentent aux élections et reçoivent chacun un numéro de 1 à 7. Après un sondage auprès d'un certain nombre d'électeurs sur leurs intentions de vote, on souhaite connaître le pourcentage estimé de voix pour chaque parti ainsi que le pourcentage d'indécis.

Écrivez le DA correspondant.

Un électeur indécis entrera un choix de 8.

Entrées :

partiChoisi
(p *)

Sorties :

pour chaque parti { numéro
pourcentVoix
(7 *)
pourcentIndécis

À créer :

tabNbVoixParti { cellule
(8 *)

Les élections

```
○-----○  
| etabliRResultatSondage |  
○-----○
```

```
  *  
  ...  
  obtenir partiChoisi  
  while (partiChoisi ≠ 0)  
  |  
  | ...  
  | obtenir partiChoisi  
  |  
  ...
```

Initialiser un tableau de 8
cellules à 0

Ajouter 1 dans la bonne cellule

Calculer et
sortir les %

Anniversaires

Sachant que l'on obtient, pour chaque étudiant de cette classe, son nom (zzz pour finir) et sa date de naissance sous la forme AAAAMMJJ, écrivez le DA permettant d'afficher le nom des étudiants nés en novembre.

Entrées :

pour chaque étudiant $\begin{cases} \text{nom} \\ \text{dateNaissance} \end{cases}$
(e *)

Sorties :

pour chaque étudiant né en Novembre {nom
(x *)

Anniversaires

Entrées :

pour chaque étudiant $\begin{cases} \text{nom} \\ \text{dateNaissance} \end{cases}$
(e *)

Sorties :

pour chaque étudiant né en Novembre {nom
(x *)

Comment stocker les étudiants de novembre ?

→ Tableau

De combien de cases ?

→ Prévoyons assez large et remplissons au fur et à mesure

Anniversaires

Tableau étudiantsNésNovembre :

Sim Pol	Pous Tom	Chac Al		
---------	----------	---------	--	--

Indice : 0 1 2 ... tailleMax-1

nbÉtudNésNov :

0	1	2	3
--------------	--------------	--------------	---

nom :

Sim Pol	Am Élie	Pous Tom	Chac Al	zzz
--------------------	--------------------	---------------------	--------------------	-----

dateNaissance :

20001125	20010511	20001125	19991105
---------------------	---------------------	---------------------	----------

Anniversaires

```
○────────────────────────────────○  
|  etablirAnnifNovembre  |  
○────────────────────────────────○  
  
  *  
  ...  
  
  obtenir nom  
  ┌ while (nom ≠ "zzz")  
  │ obtenir dateNaissance  
  │ ...  
  │ obtenir nom  
  └ ...  
  ...
```



Initialiser étudiantsNésNovembre,
nbÉtudNésNov



Calculer mois.
Si novembre, maj tableau



Sortie étudiantsNésNovembre

Afficher un tableau

```
○────────────────○ ↓ tab, taille
| afficherTableau |
○────────────────○ ↓
|
| *
| i = 0
| while (i<taille)
|   Sortir tab[taille]
|   i++
|
```

Mois festif

Sachant que l'on obtient, pour chaque étudiant de cette classe, son nom (zzz pour finir) et sa date de naissance sous la forme AAAAMMJJ, écrivez le DA permettant d'afficher le mois au court duquel il y a le plus d'anniversaire

Entrées :

pour chaque étudiant $\left\{ \begin{array}{l} \text{nom} \\ \text{dateNaissance} \end{array} \right.$
(e *)

Sorties :

moiMaxAnnif

A créer :

tabMoisAnnif

Maximum dans un tableau

Plutôt qu'un LV totalement arbitraire : prenons le 1^{er} élément

Pas besoin de stocker le max, il est dans le tableau

```
○-----○ ↓ tab, taille
|  max  |
○-----○ ↓ max, iMax

*
max = LV
iMax = -1

i = 0
while (i < taille)
    if (tab[i] > max)
        max = tab[i]
        iMax = i
    i++
```

```
○-----○ ↓ tab, taille
|  max  |
○-----○ ↓ max, iMax
// précond : taille > 0

*
max = tab[0]
iMax = 0

i = 1
while (i < taille)
    if (tab[i] > max)
        max = tab[i]
        iMax = i
    i++
```

```
○-----○ ↓ tab, taille
|  max  |
○-----○ ↓ max, iMax
// précond : taille > 0

*
iMax = 0

i = 1
while (i < taille)
    if (tab[i] > tab[iMax])
        iMax = i
    i++

max = tab[iMax]
```

Recherche dans un tableau

Un magasin de prêt-à-porter organise un concours.

Chaque client ayant fait un achat ce mois-ci a reçu un billet avec un numéro pour tenter de gagner 50€ de réduction. Le tableau `billetsGagnants` contient les 20 numéros des billets gagnants. Écrivez le module qui prend en paramètre le numéro d'un billet et le tableau et affiche la réduction gagnée par le client.

Recherche dans un tableau

Écrivez le module qui prend en paramètre le numéro d'un billet et le tableau et affiche la réduction gagnée par le client.

```
o-----o ↓ billet, billetsGagnants
| billetGagnant |
o-----o

*
iBillet = 0
while (iBillet < 20 AND billetsGagnants[iBillet] ≠ billet)
  iBillet ++

if (iBillet == 20)
  sortir "Aucune réduction"
else
  sortir "50€ de réduction"
```