Algorithmie

Support de cours formation webdesign

Frédéric Nadaradjane

Ordre du jour

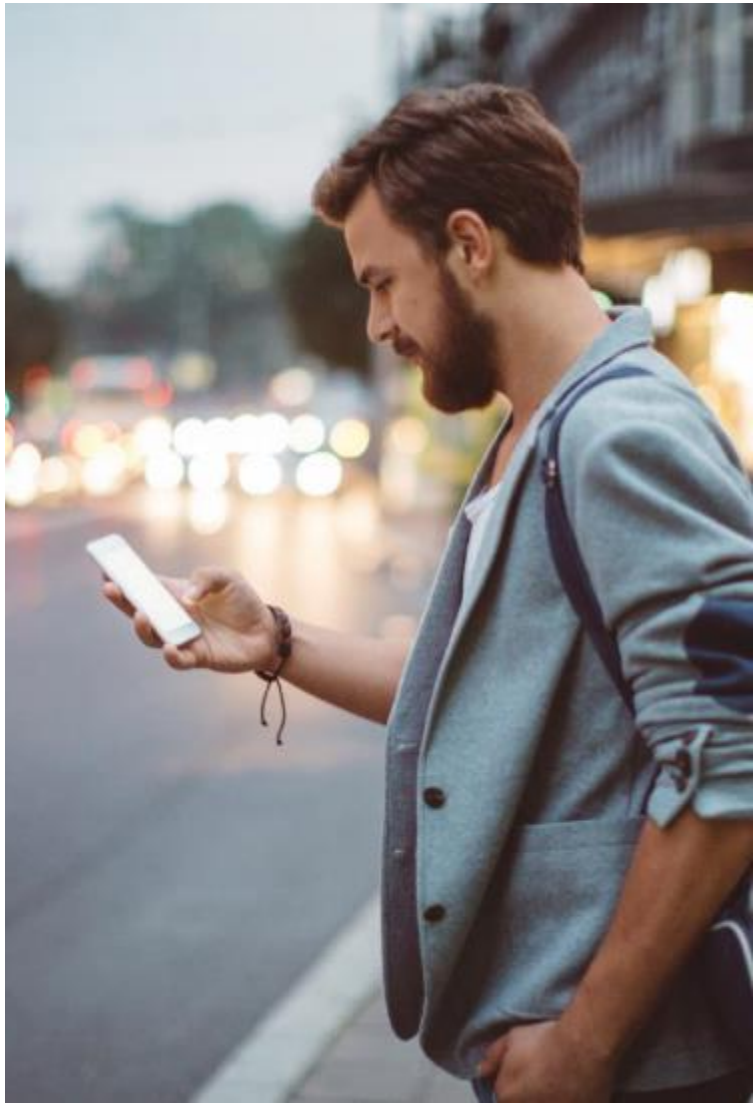
Comprendre les enjeux de l’algorithmie

Structure d’un algorithme

Introduction au pseudo code

Etapes d’un algorithme

Variables



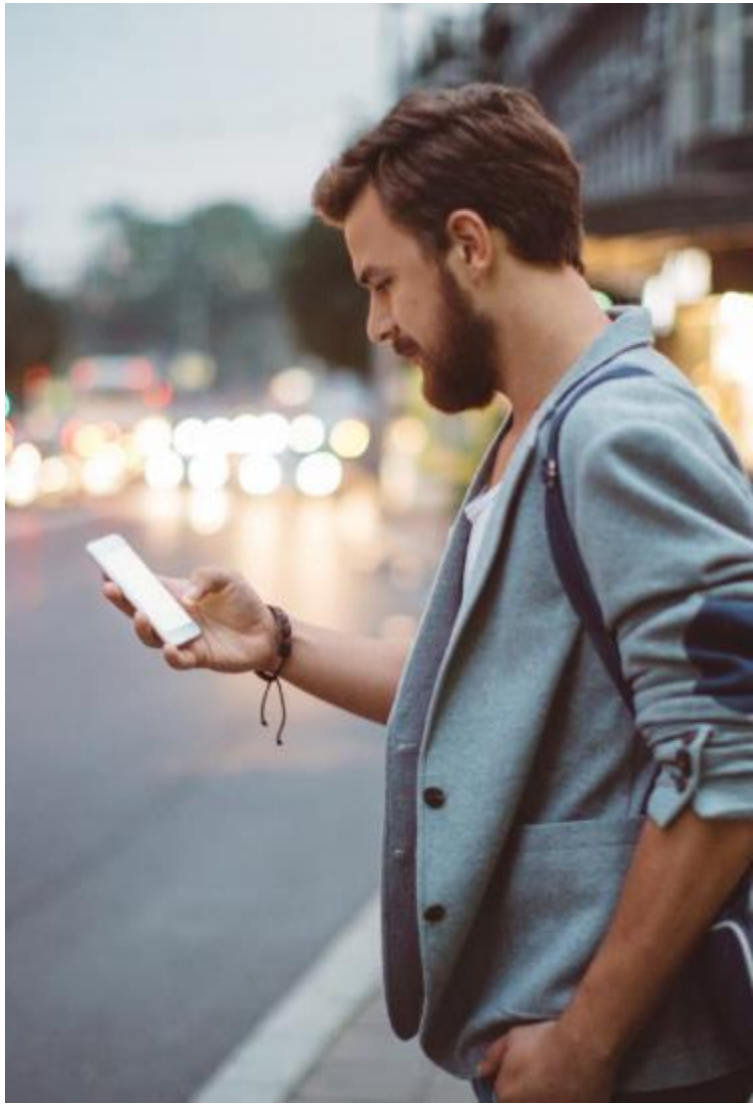
Ordre du jour

Types fondamentaux

Instructions d’entrée/sorties

Affectation

Structure de contrôle



Comprendre les enjeux de l'algorithmie

1

Comprendre les enjeux de l'algorithmie

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 ***expliciter*** son raisonnement
  - Savoir ***formaliser*** son raisonnement
  - Concevoir (et écrite) des ***algorithmes***:
    - Séquences d'instructions qui décrit comment résoudre un problème particulier

Comprendre les enjeux de l'algorithmie

Exemple d'algorithme

- Un algorithme qu'on a tous fait au moins une fois dans notre vie :

- Faire les courses :

- Si je n'ai plus de lait,
- Alors
- J'achète 3 briques de lait au magasin
- Sinon
- Je n'achète pas de lait

- Sortir avec des lunettes de soleil :
- Si il fait soleil,
- Alors
- Je sors ma meilleure paire de lunette
- Sinon
- Je sors sans lunette

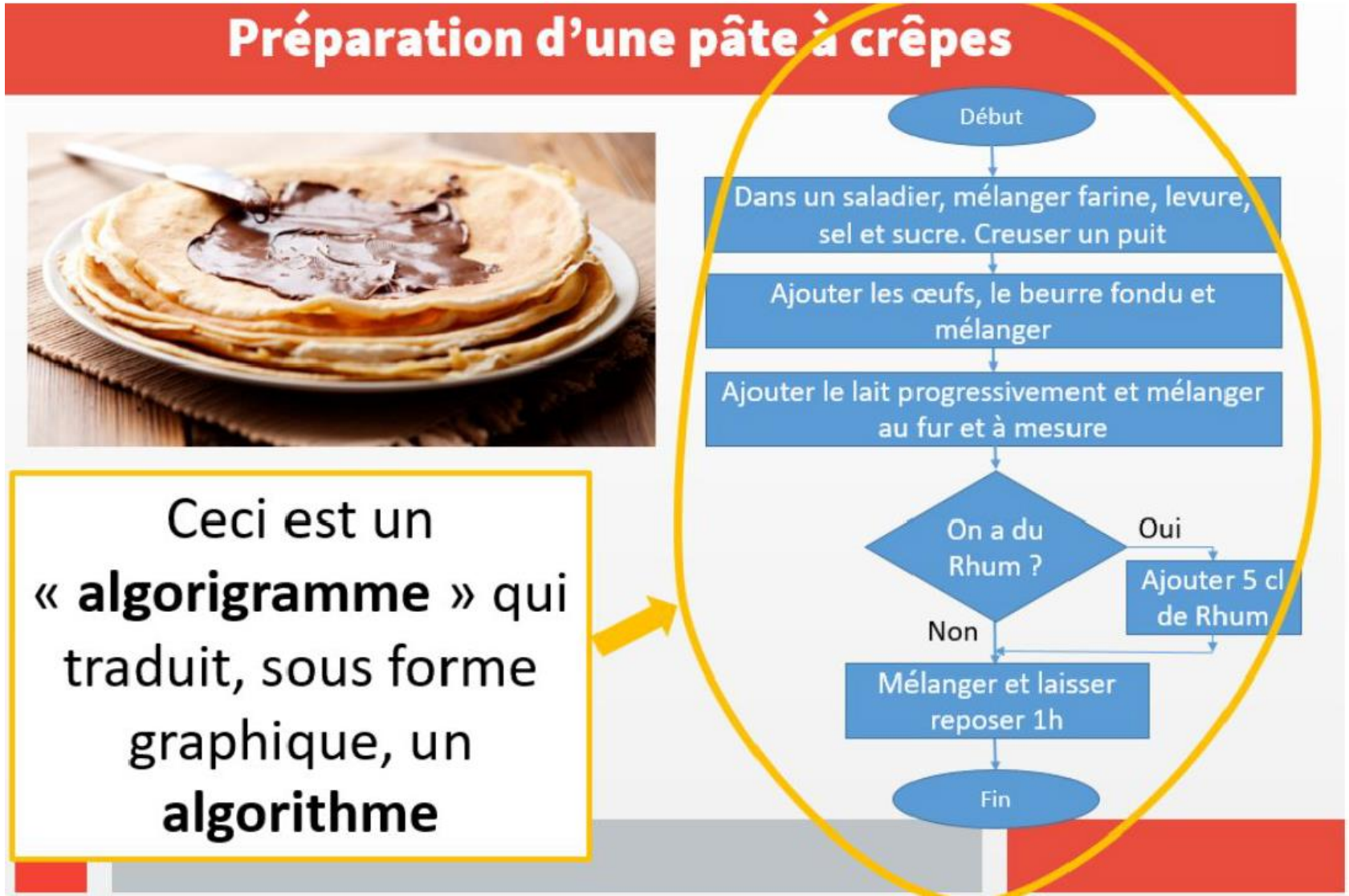
Comprendre les enjeux de l'algorithmie

Exemple d'algorithme

- D'autres exemple d'algorithmes :
- Suivre la notice de montage de meuble en kit
- Recette de cuisine

Comprendre les enjeux de l'algorithmie

Exemple d'algorithme



Comprendre les enjeux de l'algorithmie

Exemple d'algorithme

- 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 ?
  - Etant donnée une tâche, peut-on dire s'il existe un algorithme qui la résolve ?

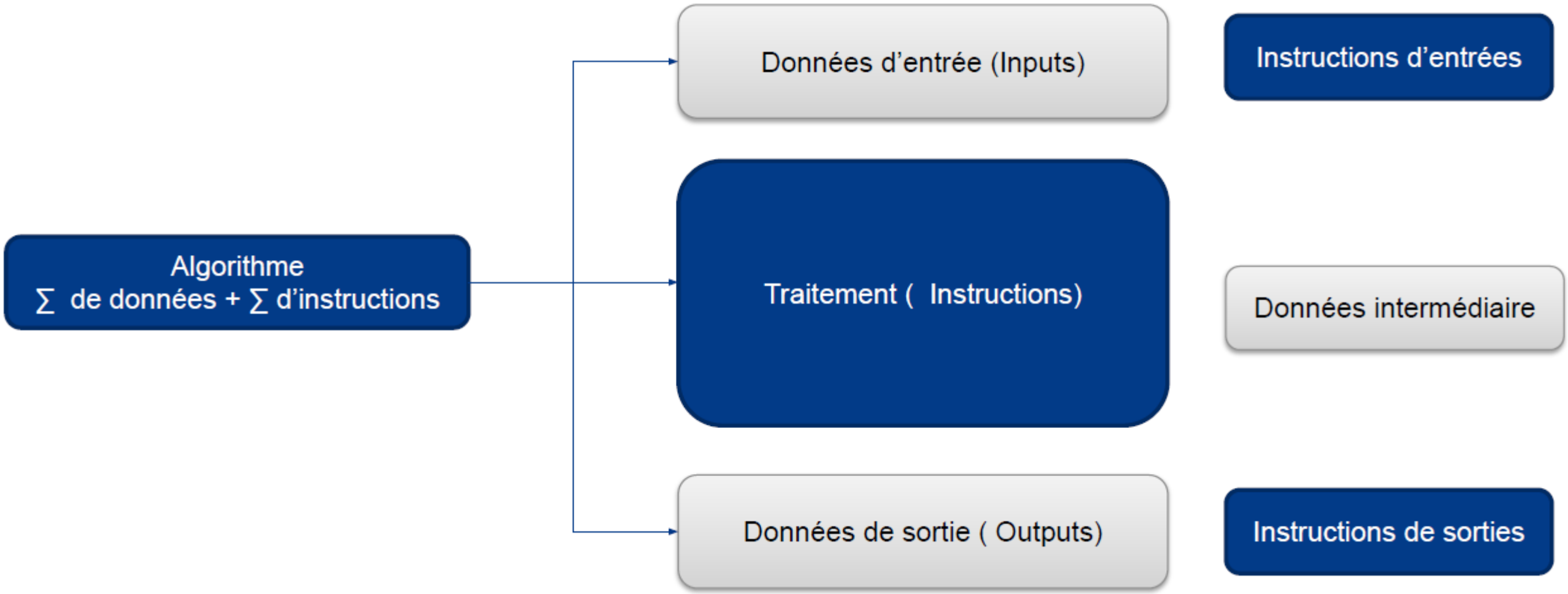
- Correction
  - Peut-on être sûr qu'un algorithme réponde au problème pour lequel il a été conçu ?

Structure d’un algorithme

2

Structure d'un algorithme

Vue globale d'un algorithme



Introduction au pseudo code

Introduction au pseudo code

A quoi sert le pseudo code

- Le pseudo code permet de traduire de façon « naturel » un algorithme sans faire appel à un langage de programmation

- L'écriture de pseudo-code permet généralement de bien comprendre la difficulté d'implémenter un algorithme et de développer des méthodes structurées dans sa construction.

Introduction au pseudo code

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.

Introduction au pseudo code

ETAPES D'UN ALGORITHME

```
Algorithme NomAlgorithme
{ ceci est un commentaire}
Début
... Actions
 Fin
```

```
Algorithme Bonjour
{il dit juste bonjour mais ... en anglais !
Début
        afficher('Hello world !!!')
        ALaLigne
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

Introduction au pseudo code

Déclaration

- **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

Introduction au pseudo code

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**

Introduction au pseudo code

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**

Introduction au pseudo code

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

Introduction au pseudo code

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

Introduction au pseudo code

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

Introduction au pseudo code

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

Introduction au pseudo code

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

Introduction au pseudo code

Algo TVA

```
Algorithme   CalculTVA
{Saisit un prix HT et affiche le prix TTC correspondant}
Constantes (TVA : réel) ←20.6
           (Titre : chaîne) ←"Résultat"
Variables  prixHT : réel
Variable prixTTC, montantTVA : réels {déclarations}
Début {préparation du traitement}
      afficher("Donnez-moi le prix hors taxe :")
      saisir(prixHT)
      prixTTC ←prixHT* (1+TVA/100)  {calcul du prix TTC}
      montantTVA ← prixTTC- prixHT
      afficher(Titre) {présentation du résultat}
      afficher(prixHT, «euros H.T. + TVA »,TVA, « devient » ,prixTTC, «eurosT.T.C. »)
Fin
```

Introduction au pseudo code

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

- On souhaites crée un algorithme qui affiche l'âge de l'utilisateur grâce à sa date de naissance

- L'âge dépend de :
- De la date du jour
- La date de naissance

Introduction au pseudo code

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

```
Algorithme calculAge
{Afficher l'age de l'utilisateur en fonction de l'année de naissance}
Constante annee_en_cours <- 2023
Variable annee_de_naissance : entier
Variable age
Début
        afficher("Entrer l'année de naissance")
        saisir(annee_de_naissance)
        age <- annee_en_cours – annee_de_naissance
        afficher(age, « ans »)
FIN
```

Introduction au pseudo code

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

```
Algorithme calculAge
{Calcul de l'âge grâce à la date de naissance}
anneesJ(entiers): 2022

anneeU(entiers)
age(entiers)
Début
        afficher(« Entrer l'année de naissance »)
        saisir(anneeU)
        age <- anneeJ-anneeU
        afficher(« Vous avez » age)
Fin
```

Introduction au pseudo code

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

❖ On souhaites crée un algorithme qui affiche l'âge de l'utilisateur grâce à sa date de naissance

```
Algorithme CalculAge
{Saisir date naissance et affiche âge}
AnneeCourrant  <- 2022
MoisCourrant <- 05
JoursCourrant <- 12

Variables AnneeNaiss, MoisNaiss, JoursNaiss : entiers
Début
        afficher(« Saisir jour naissance »)
        saisir(JoursNaiss)
        afficher(« Saisir Mois naissance »)
        saisir(MoisNaiss)
        afficher(« Saisir année naissance »)
        saisir(AnnéeNaiss)
        AnneeCourrant – AnneeNaiss
        afficher("Votre âge est : ", )
Fin
```

❖ On souhaites crée un algorithme qui affiche l'âge de l'utilisateur grâce à sa date de naissance

```
Algorithme CalculAge
{Saisir date naissance et affiche âge}
AnneeCourrant  <- 2022
MoisCourrant <- 05
JoursCourrant <- 12

Variables AnneeNaiss, MoisNaiss, JoursNaiss, age : entiers
Début
        afficher(« Saisir jour naissance »)
        saisir(JoursNaiss)
        afficher(« Saisir Mois naissance »)
        saisir(MoisNaiss)
        afficher(« Saisir Année naissance »)
        saisir(AnneeNaiss)
        age <- AnneeCourrant – AnneeNaiss
        afficher("Votre âge est : ", age)
Fin
```

```
Algorithme CalculMois
{Mois restant avant anniversaire}

MoisCourant <- 05

Variables MoisNaiss, MoisRestant : entiers
Début
        afficher(« Mois de naissance ? »)
        saisir(MoisNaiss)
        MoisRestant <- MoisNaiss – MoisCourant
        afficher (« Il vous restes  : », MoisRestant)
Fin
```

Introduction au pseudo code

Exercice

- C’est bientôt les soldes ! On souhaite calculer et afficher, à partir d’un prix initial saisi, le prix soldé

- Le montant soldé dépend de :
- Du prix
- Du pourcentage de remise

Introduction au pseudo code

Exercice

❖ Calcul solde

```
{Calcul du prix soldé}
Variable prixInitiale, remise : reels
Variable prixfinale: réels
Variable montantRemise
Début
        afficher("Entrer le prix Initiale")
        saisir(prixInitiale)
        afficher("Entrer le pourcentage de remise")
        saisir(remise)
        prixfinale <- prixInitiale * (100-remise)/100)
        montantRemise <- prixInitiale - prixfinale
        afficher(« Le prix soldé est de : », prixfinale, « le montant de la remise est de : », montantRemise)
```

Introduction au pseudo code

Exercice

❖ C’est bientôt les soldes ! On souhaite calculer et afficher, à partir d’un prix initial saisi, le prix soldé

```
Algorithme calculSolde
Variable prixInitial : réels
Variable remise : entier
Variable montantSolde, PrixAPayer : réels
Début
        afficher("prix initial ?")
        saisir(prixInitial)
        afficher("remise ?")
        saisir(remise)
        montantSolde <- prixInitial * ((100-remise)/100)
        afficher("Resultat", montantSolde)
        PrixAPayer <- prixInitial – montantSolde
        afficher(PrixAPayer)
FIN
```

Introduction au pseudo code

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)

Structure alternative

« SI ... ALORS ... SINON ... FSI »

```
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
```

Structure alternative

« SI ... ALORS ... SINON ... FSI »

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).

Structure alternative

« SI ... ALORS ... SINON ... FSI »

```
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 val ← val ×2
        fsi
        Afficher ("Voici la valeur val :" , val)
fin
```

- 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).

Structure alternative imbriquées

« SI ... ALORS ... SINON ... FSI »

Problème: afficher :

- Demander à l'utilisateur son age.

Si l'utilisateur à moins de 12 ans ils sera afficher qu'il pourra accéder au site pour enfant
Sinon l'accès sera refusé

Structure alternative imbriquées

« SI ... ALORS ... SINON ... FSI »

```
Algorithme Note
Variable moyenne : reel
        Début
                afficher(« Saisir la moyenne »)
                saisir(« moyenne »)
                si moyenne >= 12
                alors afficher ("Reçu avec mention Assez Bien ")
                sinon si moyenne > 10
                alors afficher (« passable »)
                sinon afficher (« Insuffisant »)
        FIN
```

Structure alternative imbriquées

« SI ... ALORS ... SINON ... FSI »

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
```

Structure alternative imbriquées

« SI ... ALORS ... SINON ... FSI »

Problème: afficher :

- Demander à l'utilisateur son age.

Si l'utilisateur à moins de 12 ans ils sera afficher qu'il pourra accéder au site pour enfant
Sinon l'accès sera refusé

```
Algorithme AgeRequis
Variable age
Début
        afficher(« Saisir votre age »)
        saisir(age)
        si age < 12
        alors afficher(« vous pouvez accéder au site pour enfant »)
        sinon afficher(« L'accès est refusé »)
        fsi
FIN
```

Structure alternative imbriquées

« SI ... ALORS ... SINON ... FSI »

Problème: afficher :

- Demander à l'utilisateur sa taille en cm.

Si l'utilisateur fait plus de 180cm le programme affichera « Tu est grand »
Si l'utilisateur fait plus de 160cm le programme affichera « Tu est moyen »
Si l'utilisateur fait plus de 150cm le programme affichera « Tu est petit quand même»
Dans les autres cas le programme affichera « C'est pas la taille qui compte »

Structure alternative imbriquées

« SI ... ALORS ... SINON ... FSI »

Problème: afficher :

- Demander à l'utilisateur sa taille en cm.

Si l'utilisateur fait plus de 180cm le programme affichera « Tu es grand »
Si l'utilisateur fait plus de 160cm le programme affichera « Tu es moyen »
Si l'utilisateur fait plus de 150cm le programme affichera « Tu es petit quand même»
Dans les autres cas le programme affichera « C'est pas la taille qui compte »

Structure alternative imbriquées

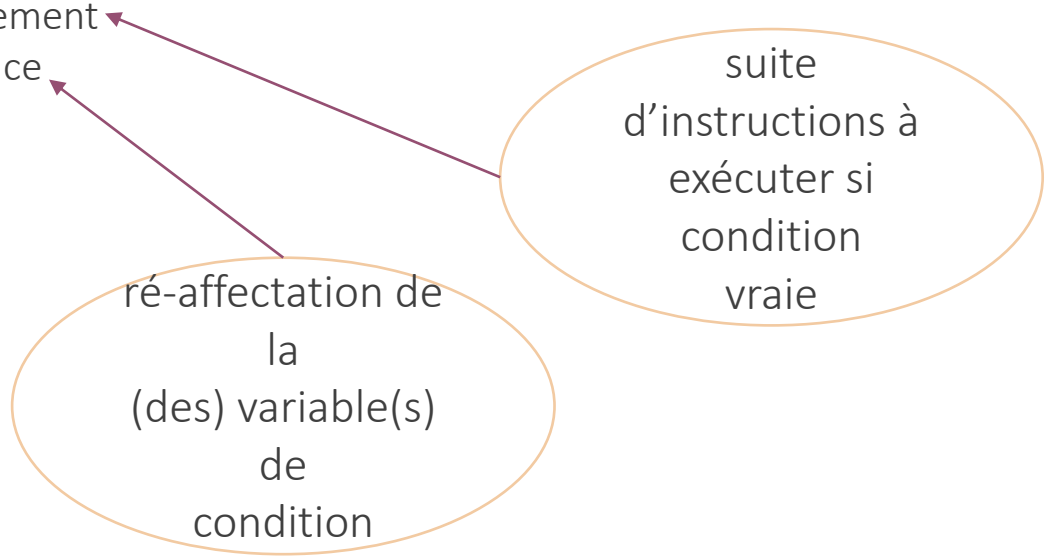
« SI ... ALORS ... SINON ... FSI »

```
Algorithme Taille
Variable taille : entier
Début
        afficher(« Saisir la taille »)
        saisir(« taille en cm»)
        si taille > 180
        alors afficher (« tu es grand »)
                sinon si taille > 160
                alors afficher (« tu es moyen »)
                        sinon si taille > 150
                        alors afficher (« tu es petit »)
                        sinon afficher(« C'est pas la taille qui compte »)
                        fsi
                fsi
        fsi
FIN
```

Les BOUCLES

TANT QUE / WHILE

```
tant que <expression logique (vraie)> faire
        traitement
        relance
ftq
```



Les BOUCLES

TANT QUE / WHILE

- 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

Les BOUCLES

TANT QUE / WHILE

Demander à l'utilisateur de ressaisir le nombre tant que le nombre saisie n'est pas 5

Algorithme nombreChance

```
Variable nbChance:entier <- 5
Variable nombre : entier
DEBUT
        afficher(« saisir nombre »)
        saisir(nombre)
        tant que nombre != nbChance faire
                afficher(« ressaisir nombre »)
                saisir(nombre)
        ftq
        afficher(« Vous avez trouvez le nombre chance qui est : », nbChance)
FIN
```

TANT QUE / WHILE

Exercice

Demander à l'utilisateur de ressaisir le nombre tant que le nombre saisie est paire

```
Algorithme NombrePaire
Variable nombre : entier
Début
        afficher(« Saisir un nombre »)
        saisir(nombre)
        tant que nombre%2 =0 faire
                afficher(« Ressaisir le nombre »)
                saisir(nombre)
        ftq
FIN
```

TANT QUE / WHILE

Exercice

Demander à l'utilisateur de ressaisir le nom tant que l'utilisateur ne s'appelle pas Tom

```
Algorithme Nom
Variable prenom : chaine de caractère
Début
        afficher(« Saisir prénom »)
        saisir(prenom)
        tant que prenom != Tom faire
        afficher(« ressaisir prénom »)
        saisir(prenom)
        ftq
FIN
```

TANT QUE / WHILE

E x e r c i c e

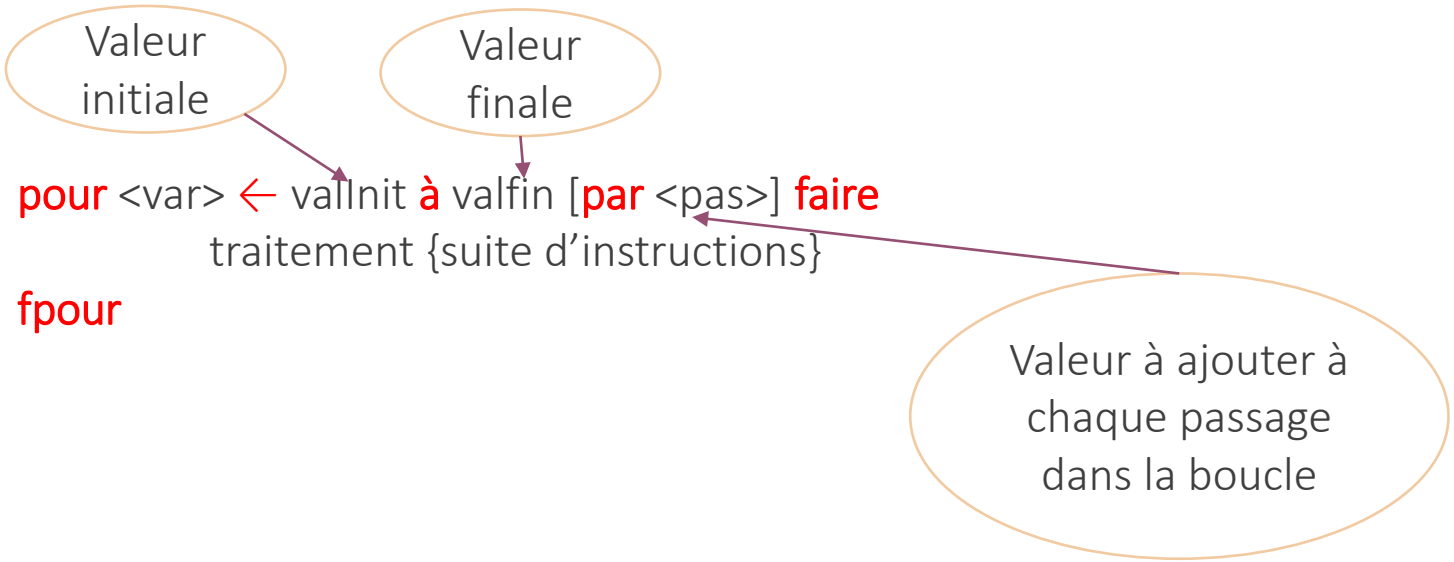
Initialiser une variable « nombreChance » à 5

Demander à l'utilisateur de ressaisir le nombre tant que le nombre saisie n'est pas 5

Algorithme nombreChance

```
Variable nbChance:entier <- 5
Variable nombre : entier
DEBUT
        afficher(« saisir nombre »)
        saisir(nombre)
        tant que nombre != nbChance faire
                afficher(« ressaisir nombre »)
                saisir(nombre)
        ftq
        afficher(« Vous avez trouvez le nombre chance qui est : », nbChance)
FIN
```

BOUCLE « POUR » ou « FOR »



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

BOUCLE « POUR » ou « FOR »

SÉMANTIQUE BOUCLE

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

BOUCLE « POUR » ou « FOR »

SÉMANTIQUE BOUCLE

```
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 ←1à nbVal faire
                afficher("Donnez une valeur :")
                saisir(valeur)
                totalValeurs←totalValeurs+ valeur   {cumul}
                Cpt <- cpt+1
        fpour
        {édition des résultats}
        afficher("Le total des ", nbVal, "valeurs est " , totalValeurs)
fin
```

BOUCLE « POUR » ou « FOR »

Exercice

L’utilisateur doit rentrer une note
Si le chiffre entrée est supérieur à 10 afficher « Bravo sur cette matière tu as eu la moyenne »
Sinon afficher « Résultat insuffisant »

Si le résultat est insuffisant le programme affichera 10 fois : « Résultat insuffisant pour un passage l’année prochaine »
Utiliser une boucle for.

BOUCLE « POUR » ou « FOR »

Exercice

```
Algorithme Note
Variable note : réels
Variable cpt : entier
Début
        afficher (« Entrer note »)
        saisir(«notes »)
        si notes > 10
        alors afficher(« Bravo ! »)
        sinon afficher résultat insuffisant et pour cpt < 10
                afficher(« Resultat insuffisant »)
                cpt <- cpt + 1
                fpour
        fsi
FIN
```

BOUCLE « POUR » ou « FOR »

E x e r c i c e

Demander à l’utilisateur de ressaisir le nombre tant que le nombre saisie n’est pas 5

Compter le nombre d’erreur

Utiliser une boucle for

```
Algo boucleMoyenne
Variables notes1, notes2, notes3, notes4, notes5, moyenne : réels
Valeur nbNotes : réels
Valeur totalNotes : réels

Début
Afficher(« saisir le nombre de notes »)
Saisir(nbNotes)
Si notes >= 10
        alors « bravo vous avez la moyenne »
        Sinon « résultat insuffisant »
Fsi

Pour notes à 5 faire
        afficher(« Donnez une note :")
        saisir(notes)
        totalNotes←totalNotes+ notes   {cumul}
Fpour
Moyenne <- totalNotes / 5

Si moyenne >= 10 alors
        afficher(« année validé »)
Sinon afficher(« année non validée »)
```