INTRODUCTION À L'ALGORITHMIQUE ET À LA PROGRAMMATION

C. BENSARI

PLAN

- NOTIONS SUR LA PROGRAMMATION
- DÉCOUVERTE DE L'ALGORITHMIQUE
- ELÉMENTS DE BASE D'UN ALGORITHME
- LES STRUCTURES CONDITIONNELLES
- LES STRUCTURES RÉPÉTITIVES
- LES TABLEAUX
- LES PROCÉDURES ET FONCTIONS
- LES CHAINES DE CARACTÈRES
- LA COMPLEXITÉ D'UN ALGORITHME

NOTIONS SUR LA PROGRAMMATION

- UN PROGRAMME INFORMATIQUE EST UNE SUITE D'INSTRUCTIONS QUI PEUVENT ÊTRE INTERPRÉTÉES PAR UNE MACHINE APRÈS DES ÉTAPES DE TRANSFORMATION DU PROGRAMME DEPUIS LE LANGAGE DE HAUT NIVEAU (HUMAIN) VERS LE LANGAGE DE BAS NIVEAU (MACHINE)
- UN PROGRAMME À LE CYCLE DE VIE SUIVANT:
  - CONCEPTION ET MODÉLISATION
  - PROGRAMMATION
  - COMPILATION
  - EXÉCUTION

CONCEPTION ET MODÉLISATION

- POUR MIEUX ÉLABORER UN PROGRAMME, IL EST NÉCESSAIRE D'ESSAYER DE SCHÉMATISER LE DÉROULEMENT DE CE DERNIER
- AVANT DE DÉFINIR UN ALGORITHME, IL FAUT PENSER À ÉLABORER UN ORGANIGRAMME/LOGIGRAMME EN AMONT
- UN ORGANIGRAMME EST NORMALISÉ ET CONSTITUÉ DE FORMES QUI ONT UNE SIGNIFICATION ET QUI SUIVENT UN ORDRE PRÉCIS (LE SENS DES FLÈCHES)
- UN PROGRAMME COMMENCE PAR UN DÉBUT ET SE TERMINE PAR UNE FIN. LA REPRÉSENTATION DE CES DEUX ÉTAPES DANS UN ORGANIGRAMME EST LA SUIVANTE :

DÉCOUVERTE DE L'ALGORITHMIQUE

- UNE DESCRIPTION D'UN TRAITEMENT AUTOMATISÉ DE DONNÉES
- LE TRAITEMENT DOIT ÊTRE RÉALISÉ SUR UN ORDINATEUR
- UN ALGORITHME DOIT ÊTRE TRADUIT DANS UN LANGAGE DE PROGRAMMATION (PASCAL, JAVA, PHP, …)
- LA MISE EN PLACE D'UN ALGORITHME DOIT ÊTRE PRÉCÉDÉE PAR UNE PHASE D'ANALYSE DU PROBLÈME À RÉSOUDRE
- LA SOLUTION DÉCRITE PAR L'ALGORITHME DOIT ÊTRE TOTALEMENT INDÉPENDANTE DU LANGAGE DE PROGRAMMATION QUI SERA UTILISÉ
- *DÉFINITION*: UN ALGORITHME EST UNE SUITE D'OPÉRATIONS ÉLÉMENTAIRES DEVANT ÊTRE EXÉCUTÉES DANS UN ORDRE DONNÉ, POUR ACCOMPLIR UNE TÂCHE DONNÉE

EXEMPLE D'UN ALGORITHME DANS LE MONDE RÉEL

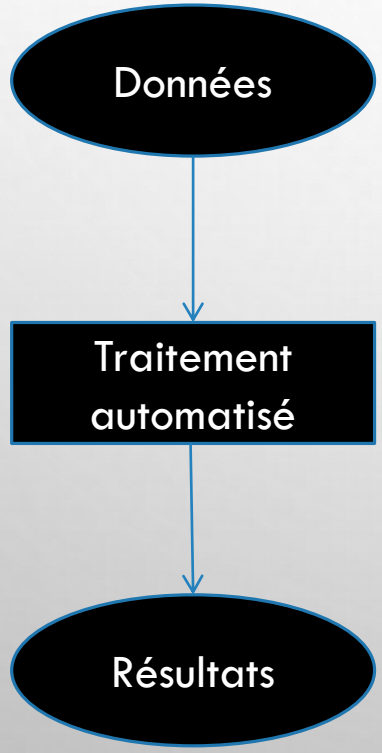
- PRÉPARATION DU RIZ :
  1- REMPLIR UNE CASSEROLE D'EAU
  2- METTRE UNE PINCÉE DE SEL
  3- METTRE LA CASSEROLE SUR LE FEU
  4- ATTENDRE JUSQU'À L'ÉBULLITION D'EAU
  5- METTRE LE RIZ DANS LA CASSEROLE
  6- LAISSER CUIRE 10 À 15 MINUTES
  7- ÉGOUTTER LE RIZ

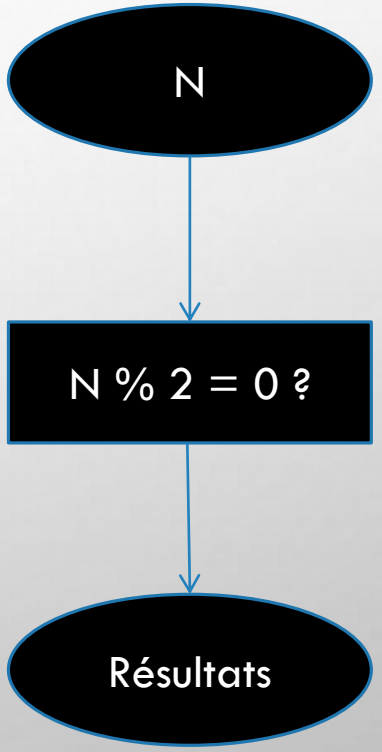
EXEMPLE D'UN PROBLÈME À RÉSOUDRE

- UN NOMBRE ENTIER N EST-IL IMPAIR ?
- SOLUTION:
  - ANALYSE: UN NOMBRE N EST IMPAIR SI LE RESTE DE LA DIVISION N PAR 2 EST ÉGALE À 1
  - SOLUTION ALGORITHMIQUE
    1. CALCULER LE RESTE R DE LA DIVISION DE N PAR 2
    2. SI R EST ÉGALE À 1 ALORS N EST IMPAIR
    3. SINON N N'EST PAS IMPAIR

PRINCIPE GÉNÉRAL

- UN TRAITEMENT AUTOMATISÉ CONSISTE À FAIRE DES OPÉRATIONS SUR DES DONNÉES (INFORMATIONS) FOURNIES EN ENTRÉE ET AFFICHER LES DONNÉES RÉSULTANTES DU TRAITEMENT EN SORTIE





ELÉMENTS DE BASE D'UN ALGORITHME

- VARIABLES
  - LES DONNÉES ET RÉSULTATS SONT DES GRANDEURS QUI SONT SUSCEPTIBLES DE VARIER => **VARIABLES**
  - LES NOMS DE VARIABLES DOIVENT COMMENCER PAR UNE LETTRE, UN UNDERSCORE « _ » OU UN DOLLAR « $ »
  - LES CONVENTIONS DE NOMMAGES PRÉCONISENT D'UTILISER LA FORME CAMELCASE (PREMIÈRE LETTRE EN MINUSCULE ET CHAQUE NOUVEAU MOT DU NOM DOIT COMMENCER PAR UNE MAJUSCULE : MAMAGNIFIQUEVARIABLE)

    EXEMPLES:

    VARIABLES

    RAYONCERCLE, SURFACECERCLE : RÉELS

- TYPES
  - LES NUMÉRIQUES: LES NOMBRES ENTIERS ET RÉELS (EX. 3, -7, 1.25, 1 E3)
  - LES CHAINES DE CARACTÈRES (EX. " BONJOUR")
  - LE TYPE BOOLÉEN (DEUX VALEURS POSSIBLE: VRAI OU FAUX)

ELÉMENTS DE BASE D'UN ALGORITHME

- LES CONSTANTES :
  - CERTAINS TYPES DE DONNÉES (INFORMATIONS) NE SONT PAS AMENÉES À CHANGER PENDANT TOUT LE PROGRAMME => **CONSTANTES**
  - LES CONVENTIONS DE NOMMAGE PRÉCONISENT D'UTILISER DES MAJUSCULES ET DES UNDERSCORES POUR LE NOM DE CONSTANTES

EXEMPLES:

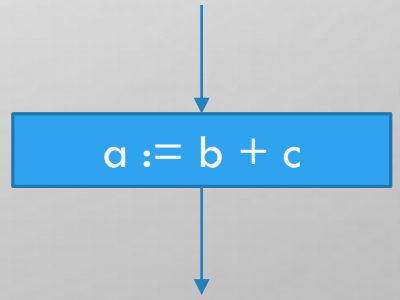
CONSTANTES

PI = 3,14159

NB_JOURS_DE_SEMAINE = 7

ELÉMENTS DE BASE D'UN ALGORITHME

- EXPRESSIONS ET AFFECTATIONS
  - UN ALGORITHME EST UNE SUITE D'OPÉRATIONS (**INSTRUCTIONS**). DE MANIÈRE GÉNÉRALE IL S'AGIT D'ÉVALUER UNE **EXPRESSION** COMPORTANT :
    - DES VARIABLES ET DES CONSTANTES
    - DES OPÉRATIONS : +, -, *, /, >, <, >=, <=, =, <>,!=, % (%: RESTE DE LA DIVISION)
    - DES FONCTIONS PLUS COMPLEXES
  - LES RÉSULTATS DE L'ÉVALUATION DE CES EXPRESSIONS SONT GÉNÉRALEMENT « RANGÉES » DANS DES VARIABLES. ON DIT QU'IL S'AGIT D'UNE AFFECTATION DU RÉSULTAT D'UNE EXPRESSION À UNE VARIABLE.
  - POUR L'AFFECTATION ON UTILISE LE SYMBOLE «:=» OU « ← »
  - LES EXPRESSIONS ET LES AFFECTATIONS SONT REPRÉSENTÉES DANS UN ORGANIGRAMME AVEC LE SYMBOLE RECTANGLE :



ELÉMENTS DE BASE D'UN ALGORITHME

- EXEMPLE D'UNE AFFECTATION SIMPLE:

VARIABLES

A, B, C : **ENTIER**;

A:= 5; % AFFECTER LE NOMBRE 5 À LA VARIABLE **A** %

B:=10;

- EXEMPLE D'UNE AFFECTATION AVEC UNE EXPRESSION :

VARIABLES

RAYON, SURFACE : RÉELS;

CONSTANTES

PI := 3.14;

SURFACE := PI * RAYON * RAYON; % AFFECTATION DU RÉSULTAT DE L'EXPRESSION À LA VARIABLE SURFACE %

ELÉMENTS DE BASE D'UN ALGORITHME

- ATTENTION AUX RÈGLES DE PRIORITÉ :

  1- LES FONCTIONS MATHÉMATIQUES EN PREMIER

  2- LA MULTIPLICATION ET LA DIVISION

  3- L'ADDITION ET LA SOUSTRACTION

  EXEMPLE :

  % SQR EST UNE FONCTION MATHÉMATIQUE CALCULANT LA RACINE CARRÉ D'UN NOMBRE %

  C := SQR(A) + 5 * B;

  ANALYSE :

  POUR **A = 4** ET **B = 3, C** SERA ÉGALE À **17** ET NON PAS **21**

ELÉMENTS DE BASE D'UN ALGORITHME

- OPÉRATIONS D'ENTRÉE/SORTIE :

  DANS N'IMPORTE QUEL PROGRAMME, UN ÉCHANGE A LIEU ENTRE L'UTILISATEUR ET LA MACHINE :

  LA SAISIE PAR CLAVIER → DONNÉES D'ENTRÉE

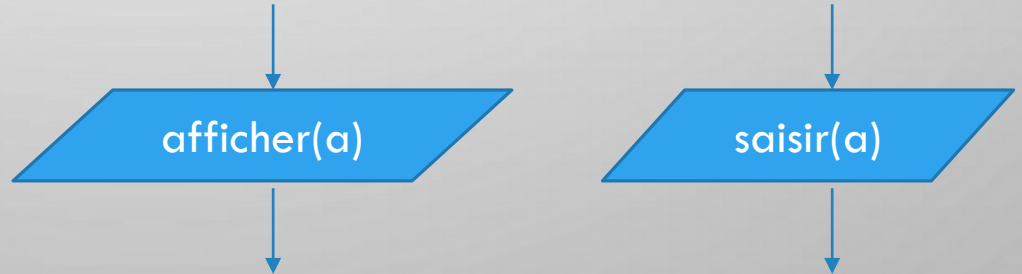
  AFFICHAGE SUR ÉCRAN → DONNÉES EN SORTIE

  EXEMPLE : CALCUL DE LA SURFACE D'UN CERCLE

  - SAISIR LA VALEUR DU RAYON DU CERCLE (DONNÉE D'ENTRÉE OU LECTURE)
  - AFFECTER À UNE VARIABLE **SURFACE** LE RÉSULTAT DE L'EXPRESSION $\Pi*(\text{RAYON})^2$
  - AFFICHER LE RÉSULTAT (DONNÉE DE SORTIE OU ÉCRITURE)

  → CETTE SUITE D'OPÉRATIONS EST APPELÉ « **SÉQUENCE D'INSTRUCTIONS** »

- LES OPÉRATIONS D'ENTRÉ/SORTIE SONT REPRÉSENTÉES DANS UN ORGANIGRAMME AVEC UN RECTANGLE INCLINÉ



ORGANIGRAMME DE CALCUL DE LA SURFACE D'UN CERCLE

SYNTAXE GÉNÉRALE D'UN ALGORITHME

ALGORITHME CALCUL_SURFACE_CERCLE

CONSTANTES

PI := 3.14;

VARIABLES

RAYON, SURFACE : **REELS;**

DEBUT

AFFICHER("ENTREZ LA VALEUR DU RAYON");

SAISIR(RAYON);

SURFACE := PI * RAYON * RAYON;

AFFICHER(SURFACE);

FIN

LES STRUCTURES CONDITIONNELLES

- UNE STRUCTURE CONDITIONNELLE PERMET DE FAIRE UN TRAITEMENT SELON UNE OU PLUSIEURS CONDITIONS
- STRUCTURE CONDITIONNELLE SIMPLE
  - LA STRUCTURE CONDITIONNELLE SIMPLE SE PRÉSENTE SOUS LA FORME SUIVANTE :

SI (CONDITION) **ALORS DÉBUT**

/*SÉQUENCE D'INSTRUCTIONS*/

FIN

FINSI

EXEMPLE :

VARIABLES

A : **BOOLÉEN;**

B : **ENTIER;**

SAISIR(B);

A := B > 5;

SI (A) **ALORS** /* FONCTIONNERA AUSSI AVEC : **SI** (B>5) */

AFFICHER("LA VALEUR DE A EST SUPÉRIEURE À 5");

FINSI

LES STRUCTURES CONDITIONNELLES

- LA STRUCTURE CONDITIONNELLE COMPOSÉE SE PRÉSENTE SOUS LA FORME SUIVANTE :

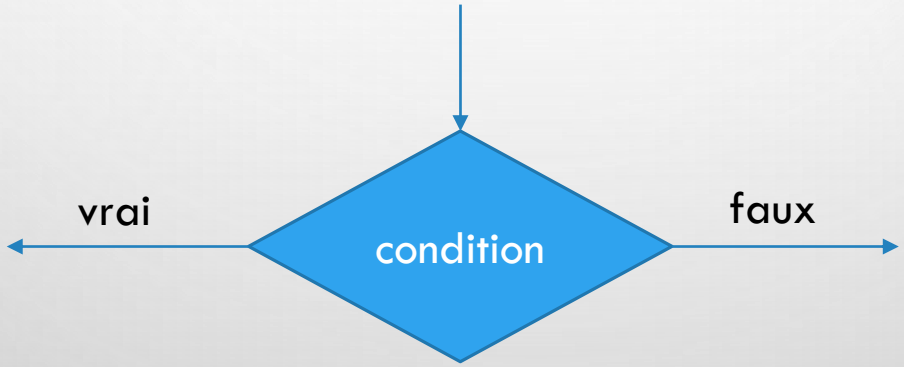
```
SI (CONDITION) ALORS DÉBUT
                    /*SÉQUENCE D'INSTRUCTIONS*/
                    FIN
               SINON DÉBUT
                    /*SÉQUENCE D'INSTRUCTIONS*/
                    FIN
FINSI
```

EXEMPLE :

```
VARIABLES
        A : BOOLÉEN;
        B : ENTIER;
SAISIR(B);
SI (B > 5) ALORS   AFFICHER("LA VALEUR DA EST SUPÉRIEURE À 5");
        SINON AFFICHER("LA VALEUR DA EST INFÉRIEURE OU ÉGALE À 5");
FINSI
```

LES STRUCTURES CONDITIONNELLES

- DANS UN ORGANIGRAMME, LA STRUCTURE CONDITIONNELLE EST REPRÉSENTÉE AVEC LA FORME SUIVANTE :



LES STRUCTURES CONDITIONNELLES

- **STRUCTURE CONDITIONNELLE MULTIPLE**
  - DANS UNE STRUCTURE CONDITIONNELLE MULTIPLE ON PEUT COMPARER NOTRE OBJET (VARIABLE OU EXPRESSION) AVEC TOUTE UNE SÉRIE DE VALEURS ET D'EXÉCUTER UN ENSEMBLE D'INSTRUCTIONS EN FONCTION DE LA VALEUR EFFECTIVE
  - UNE SÉQUENCE PAR DÉFAUT PEUT ÊTRE PRÉVUE DANS LE CAS OÙ L'OBJET N'EST ÉGALE À AUCUNE DES VALEURS ÉNUMÉRÉES
  - SYNTAXE D'UNE STRUCTURE CONDITIONNELLE MULTIPLE:

```
SUIVANT [VARIABLE_OU_EXPRESSION] FAIRE
        VALEUR_1 : /* INSTRUCTIONS*/
        VALEUR_2 : /* INSTRUCTIONS*/
        VALEUR_3 : /* INSTRUCTIONS*/
        SINON /* INSTRUCTIONS PAR DÉFAUT */
FIN_SUIVANT
```

LES STRUCTURES CONDITIONNELLES

- EXEMPLE D'UTILISATION D'UNE STRUCTURE CONDITIONNELLE MULTIPLE :

```
…..
SAISIR(JOUR);
SUIVANT JOUR FAIRE
        ”SAMEDI” : AFFICHER(” C’EST LE WEEKEND !”);
        ”DIMANCHE” : AFFICHER(” C’EST LE WEEKEND !”);
         SINON AFFICHER(” CE N’EST PAS LE WEEKEND !”);
FIN_SUIVANT
……
```

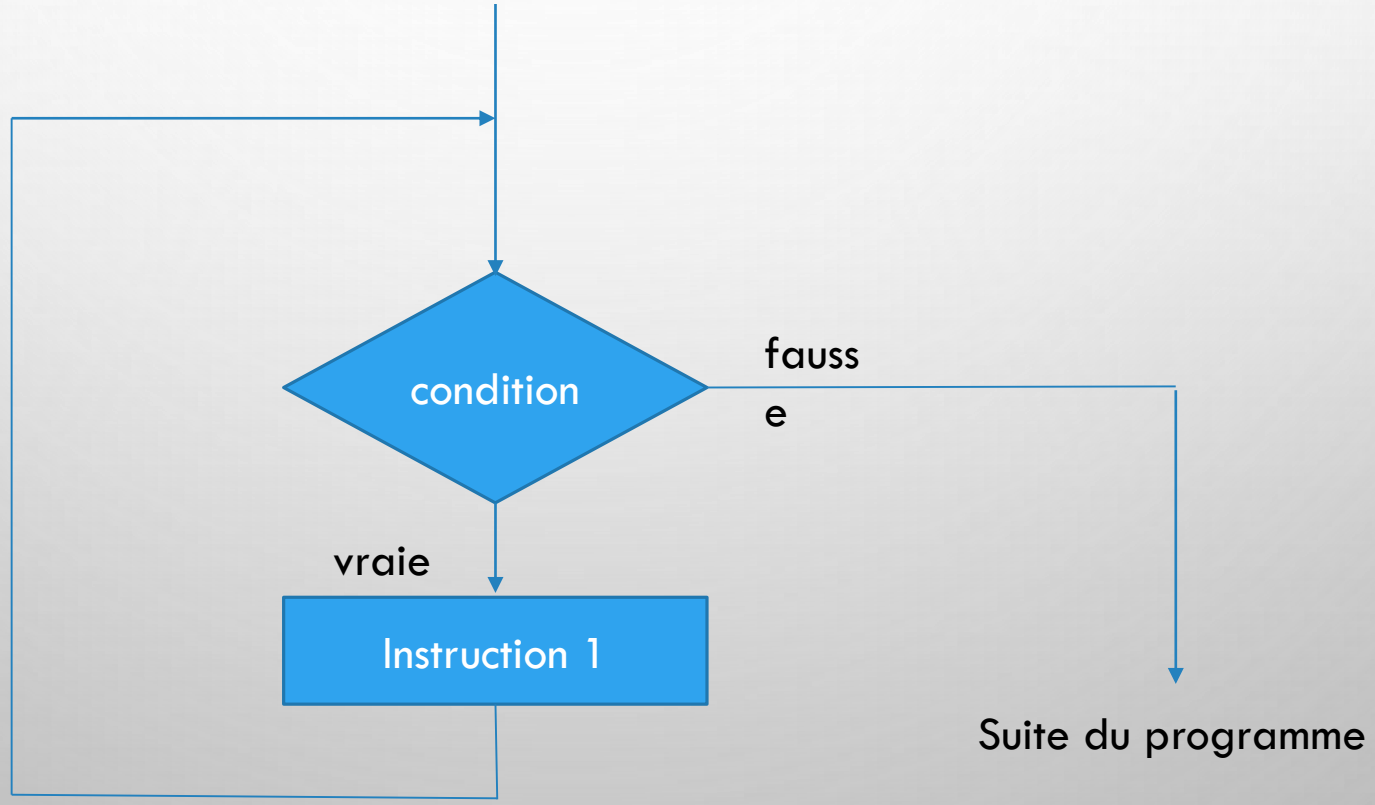
LES STRUCTURES RÉPÉTITIVES (BOUCLES)

- LES STRUCTURES RÉPÉTITIVES SONT UTILISÉES LORSQUE UNE OU PLUSIEURS INSTRUCTIONS DOIVENT ÊTRE EXÉCUTÉES PLUSIEURS FOIS
- TROIS ÉLÉMENTS NÉCESSAIRES DANS UNE STRUCTURE RÉPÉTITIVE:
  - L'INITIALISATION : PRÉCISE LE POINT DE DÉPART DE LA BOUCLE
  - CONDITION D'ARRÊT DE LA BOUCLE : VÉRIFIE SI LA BOUCLE DOIT CONTINUER À RÉPÉTER LE TRAITEMENT
  - L'INCRÉMENTATION: ELLE CHANGE LA VALEUR DE LA VARIABLE UTILISÉE DANS LA CONDITION D'ARRÊT
- TROIS TYPES DE BOUCLES UTILISABLES EN PROGRAMMATION :
  - LA BOUCLE **TANT QUE … FAIRE** :

```
TANT QUE (CONDITION) FAIRE
        /* INSTRUCTIONS DONT L'INCRÉMENTATION*/
FIN_TANT_QUE
```

LES STRUCTURES RÉPÉTITIVES (BOUCLES)

- LA REPRÉSENTATION DANS UN ORGANIGRAMME D'UNE STRUCTURE RÉPÉTITIVE EST FAITE DE LA MANIÈRE SUIVANTE :



LES STRUCTURES RÉPÉTITIVES (BOUCLES)

- EXEMPLE D'UTILISATION DE LA BOUCLE **TANT QUE … FAIRE**

  - ALGORITHME PERMETTANT D'AFFICHER LES NOMBRE ENTIERS DE 1 À 10

```
ALGORITHME TANT_QUE
VARIABLES
        CONTINUER :BOOLÉEN;
DÉBUT
        CONTINUER := VRAI;        /* INITIALISATION */
        TANT QUE (CONTINUER) FAIRE        /* CONDITION D'ARRÊT */
                AFFICHER("BONJOUR");
                SI (RAND()  > 0.5){
                        CONTINUER := FAUX; /* INCRÉMENTATION */
                }
        FIN_TANT_QUE
FIN
```

LES STRUCTURES RÉPÉTITIVES (BOUCLES)

- LA STRUCTURE **RÉPÉTER … JUSQU'À**
  - SYNTAXE DE LA BOUCLE **RÉPÉTER … JUSQU'À** :

```
RÉPÉTER
        /* INSTRUCTIONS */
JUSQU'À (CONDITION)
```

EXEMPLE :

```
I:= 20;
RÉPÉTER
        ECRIRE(I);
        I--;
JUSQU'À (I = 10)
```

- A LA DIFFÉRENCE DE LA STRUCTURE **TANT QUE**, LA STRUCTURE **RÉPÉTER** POSITIONNE LA CONDITION D'ARRÊT À LA FIN DE LA BOUCLE ET NON PAS AU DÉBUT. DANS CE CAS, LES INSTRUCTIONS À L'INTÉRIEUR DU **BLOC** DE LA BOUCLE SONT EXÉCUTÉES **AU MOINS UNE FOIS**

LES STRUCTURES RÉPÉTITIVES (BOUCLES)

- LA STRUCTURE **POUR … FAIRE**
  - CETTE STRUCTURE PERMET DE RÉPÉTER UNE SÉQUENCE UN NOMBRE CONNU DE FOIS
  - SYNTAXE DE LA BOUCLE **POUR … FAIRE** :

```
POUR (INITIALISATION;  TEST_D'ARRÊT;  INCRÉMENTATION) FAIRE
        /* INSTRUCTIONS */
FIN_POUR;
```

EXEMPLE :

```
POUR (I:=1; I <= 10; I++) FAIRE
        AFFICHER(I);
FIN_POUR;
```

LES STRUCTURES RÉPÉTITIVES (BOUCLES)

- LES BOUCLES IMBRIQUÉES
  - IL EST POSSIBLE D'IMBRIQUER DES BOUCLES LES UNE DANS LES AUTRES. UNE BOUCLE **TANT QUE** PEUT CONTENIR UNE AUTRE BOUCLE **TANT QUE** COMME ELLE PEUT CONTENIR AUSSI TOUT AUTRE TYPE DE BOUCLES.
  - EXEMPLE (AFFICHERA : 4, 8, 5, 10, 6, 12, 7 ET 8)

```
POUR (I:=1; I<= 5; I++) FAIRE
        SOMME := 0;
        POUR (J:=1; J<= 2; J++) FAIRE
                SOMME : = SOMME + I +3;
                AFFICHER(SOMME);
        FIN_POUR;
FIN_POUR;
```

LES TABLEAUX

- UN ENSEMBLE DE DONNÉES DU MÊME TYPE
- CHAQUE VALEUR DU TABLEAU POSSÈDE UN INDICE (POSITION DANS LE TABLEAU). LA PREMIÈRE VALEUR DU TABLEAU SE TROUVE À L'INDICE 0
- EXEMPLE DE DÉCLARATION D'UN TABLEAU:

NOTES(0, 9) : TABLEAU DE RÉELS

NOTES: NOM DU TABLEAU (VARIABLE)

0: PREMIER INDICE DU TABLEAU

9: DERNIER INDICE DU TABLEAU

NOTES

- DANS UN ALGORITHME, LE NOM D'UN TABLEAU N'EST JAMAIS ÉCRIS TOUT SEUL, IL EST TOUJOURS SUIVI D'UN INDICE ENTOURÉ DE PARENTHÈSES :
  - ECRIRE(NOTES(1)); → AFFICHERA 10
  - NOTES(3) := 15; → AFFECTATION DE LA VALEUR 15 À LA CASE 3 DU TABLEAU NOTES

CRÉATION D'UN TABLEAU

ALGORITHME SAISIR_TABLEAU

VARIABLES TABLEAU(1 : 10) : **TABLEAU D'ENTIERS**

I: **ENTIER**

DEBUT

POUR I:=1 **À** 10 **FAIRE**

AFFICHER('ENTREZ UN NOMBRE : ');

SAISIR(TABLEAU(I)); -- > PERMET DE REMPLIR LA CASE I DE TABLEAU

AFFICHER(TABLEAU(I)); -->PERMET D'AFFICHER LE CONTENUE DE LA CASE I

FIN_POUR

FIN

AFFICHAGE D'UN TABLEAU

- POUR AFFICHER UN TABLEAU, IL FAUT PARCOURIR LE TABLEAU ÉLÉMENT PAR ÉLÉMENT EN FAISANT VARIER L'INDICE ET AFFICHER LE CONTENU DE CHAQUE CELLULE

```
DÉBUT
    POUR I:=1 À 10 FAIRE
            AFFICHER(NOMBRES(I));   /*PERMET D'AFFICHER LE CONTENU DE LA CASE I */
    FIN_POUR
FIN
```

RECHERCHER LE PLUS PETIT ÉLÉMENT DANS UN TABLEAU

- POUR DÉTERMINER LE PLUS PETIT ÉLÉMENT D'UN TABLEAU IL FAUT SUIVRE LE PRINCIPE SUIVANT :
  - SUPPOSER QUE LA PREMIÈRE CASE DU TABLEAU CONTIENT LE MINIMUM
  - COMPARER LE CONTENU DE LA DEUXIÈME CASE AVEC LE MINIMUM. SI CELUI-CI EST INFÉRIEUR, IL DEVIENT LE MINIMUM
  - RENOUVELER LA MÊME OPÉRATION POUR LE RESTE DES CASES

TRIER UN TABLEAU D'ENTIERS

- UN TABLEAU PEUT ÊTRE TRIÉ PAR ORDRE CROISSANT/DÉCROISSANT SI LE CONTENU D'UNE CASE « I » EST INFÉRIEUR/SUPÉRIEUR AU CONTENU DE LA CASE « I+1 »
- L'UNE DES MANIÈRES POUR TRIER UN TABLEAU SE FAIT SELON LE PRINCIPE SUIVANT :
  - RECHERCHER LE PLUS PETIT/GRAND ÉLÉMENT DU TABLEAU ET L'AMENER À LA PREMIÈRE CASE
  - RÉPÉTER L'OPÉRATION POUR AMENER LE MINIMUM/MAXIMUM DES CASES RESTANTES À LA DEUXIÈME CASE ET AINSI DE SUITE
- POUR TRIER UN TABLEAU IL FAUT PASSER PAR DEUX BOUCLES IMBRIQUÉES L'UNE DANS L'AUTRE
- ATTENTION AUX PERTES D'INFORMATIONS LORS DES DÉPLACEMENTS DES ÉLÉMENTS !