

Département Mathématiques Et Informatique

Algorithmique & Programmation Langage C

Chapitre 2 : Concepts de base

Objectifs

- ➡ **Comprendre les bases et concepts algorithmiques et leurs notations :**
 - **Types de données,**
 - **Les variables et constantes,**
 - **Les instructions primitives,**
 - **Les structures de contrôles.**

- ➡ **Apprendre à formuler des algorithmes simples et à les coder en langage C**

Chap. II : Les Concepts de base

- Les Types de données,
- Notion d'objets(Constantes et de variables)
- Notion d'instructions primitives
- Notion d'opérateurs et d'expressions
- Les Structures de contrôle
 - Séquence,
 - Test,
 - Boucles.

I. Notion de types de données

1. But

Le type d'un objet permet de savoir :

- ✍ Le **domaine** des **valeurs** pouvant être prises par un objet
- ✍ La **taille** de l'espace nécessaire pour conserver la donnée d'un objet de ce type (**nombre d'octets** à réserver en mémoire)
- ✍ l'ensemble des **opérateurs** qui peuvent être appliqués à la donnée

2. Classification

On distingue deux catégories de types :

- Types **simples** : représentent des données simples (**entiers, réels, caractères, logiques, Pointeurs**)
- Types **composés (construits)** : définis à partir des types simples pour **représenter des données complexes : tableaux** (vecteurs, matrices), **chaîne de caractères , structures ou enregistrements ...)**

3. Les types numériques

Représentent les domaines usuels fournis par les mathématiques : **Entier** et **Réel**. Les nombres de ces types sont utilisés dans opérations **arithmétiques** et **relationnelles**.

Les opérateurs arithmétiques

Opérateur	Signification
+, -, *	Addition, Soustraction , Multiplication
/	Division exacte
Div	Division entière
Mod	Reste de la division entière

Les opérateurs Relationnels

Opérateur	Signification
=, <>	Égal, Différent
<, <=	Inférieur, Inférieur ou égal
> , >=	Supérieur, Supérieur ou égal

Les Fonctions de base

Fonction	Signification
abs(x)	Valeur absolue
racine(x)	Racine carrée
cos(x), sin(x), atan(x)	Cosinus, Sinus et Arc tangente
Exp(x), Ln(x)	Exponentielle, Logarithme népérien

Représentation des constantes

Entiers :

-10 , 20 , -512

Réels :

0.05 , -0.5 e -4 ,
1.25 e +10 , 1. e-6

4. Le type Logique

Représente des entités logiques (caractérisées par les deux états **Vrai/Faux** ou **Oui/Non**). Les données de ce type sont utilisées dans des expressions logiques avec les opérateurs logiques **Non**, **Et**, **Ou**, **Xou**

Tables de vérité

	Non
Vrai	Faux
Faux	Vrai

ET	Vrai	Faux
Vrai	Vrai	Faux
Faux	Faux	Faux

OU	Vrai	Faux
Vrai	Vrai	Vrai
Faux	Vrai	Faux

XOU	Vrai	Faux
Vrai	Faux	Vrai
Faux	Vrai	Faux

5. Le type caractère

à chaque caractère est associé un nombre compris entre 0 et 255 appelé code **ASCII**.

L'ensemble des caractères est constitué par :

- Les Caractères de contrôle (non imprimables) : codes 0 .. 32
- Les chiffres 0 .. 9 : codes 48..57
- Les 26 lettres majuscules A..Z(minuscules a..z) :codes 65..90 (97 122)
- Les caractères spéciaux (+ - * / . : { } () ...

une constante caractère est représentée entre apostrophes : **'A'** , **'a'** , **'1'** , **'2'**

Les opérateurs et fonctions sur les caractères

Opérateurs relationnels	= , <> , < , <= , > , >=
Fonctions	ord (caractère) : Donne le code Ascii du caractère car (entier) : Donne le caractère d'un code Ascii

6. Le type chaîne

Le type chaîne permet de définir des objets composés d'une liste de caractères adjacents(un mot, une phrase).

On distingue deux utilisations :

- Chaîne de longueur variable : **Chaîne**
- Chaîne de longueur fixe n (la longueur de l'objet chaîne ne peut dépasser n caractères) : **Chaîne[n]**

Une chaîne constante littérale est représentée entre guillemets " " .

Exemple :

"la date de rentrée scolaire est 01/09"

Fonctions sur les chaînes

Fonctions	Description
Longueur(x)	Renvoie la longueur de la chaîne x
Copier(x,y)	Copie le contenu de la chaîne y vers x
Concat(x,y)	Concatène le contenu de y à celui de x
Comparer(x,y)	Compare x à y et renvoie 0 si égalité, -1 si x<y et 1 sinon
...	...

7. Priorité des opérateurs (précédence)

Priorité	Opérateur	Ordre
1 (la plus forte)	() []	→
2 (unaire)	- NOT	←
3	* / DIV MOD	→
4	+ -	→
5	< <= > >=	→
6	= <>	→
7	ET	→
8	OU	→

II. Notion d'objet

Un objet est une **entité d'information** sur laquelle on peut appliquer certaines opérations pour transformer son **état**. Un objet est caractérisé par :

- Un **nom** : Identificateur de l'objet qui représente le **contenant** de l'état
- Un **type** : nature de l'objet
- Une **Valeur** : état de l'objet qui constitue le **contenu**

1. Classification des objets

- ☞ Selon la **variation** de la valeur, Il y a deux sortes d'objets :
- Les **variables** : leurs états peuvent être modifiés par les opérations (changement d'état dans le temps)
 - Les **constantes** : leurs états ne peuvent pas être modifiés par les opérations

Exemples

Variables :

R, P, S (variables réelles)

n1, n2, n3 (variables entières)

Constantes : Pi

Chap II : Les Concepts de base

☞ Selon la **structure** de la valeur, Il y a deux sortes d'objets :

- Les **objet simples** : leurs états sont composés d'une valeur non décomposable (**atomique**).

Exemples : entier, réel, caractère

- Les **objets composés** : leurs états sont composés de plusieurs valeurs :
 - Les **tableaux** : plusieurs valeurs de même type.

Exemples : liste de notes, liste de températures, liste d'étudiants

- Les **enregistrements** : plusieurs valeurs de types différents .

Exemples : Fiche étudiant, Fiche produit, Fiche cours

2. Représentation

Physiquement un objet est représenté par un **espace mémoire** :

Nom de objet → **adresse** de cet espace (contenant)

Valeur de l'objet → le **contenu** de cet espace

Type de l'objet → la **taille** de cet espace



Contenant ≡ **Identificateur** ≡ **Adresse**

Remarques

- Le nom et le type d'un objet sont **fixes** (ne changent pas).
- On doit toujours **déclarer** un objet avant de l'utiliser
- un objet variable peut être un objet **d'entrée(donnée)**, un objet **intermédiaire** ou un objet **de sortie(résultat)**

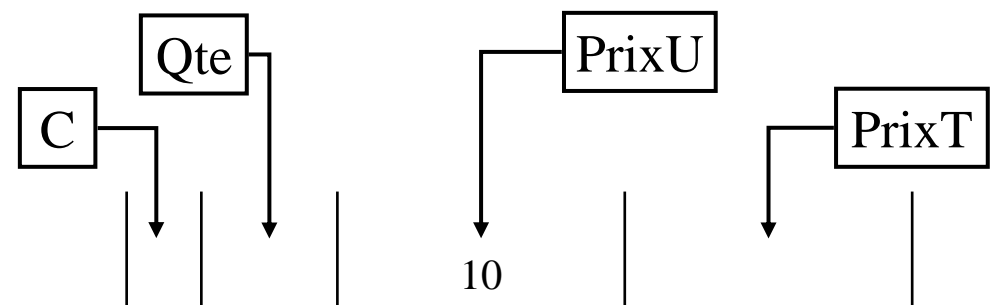
Déclaration de variables

Nom_type nom_var1[=val_Init] [, nom_var2[=val_Init], ...]

Exemples

Caractère	C
Entier	Qte =10
Réel	PrixU , PrixT

Représentation



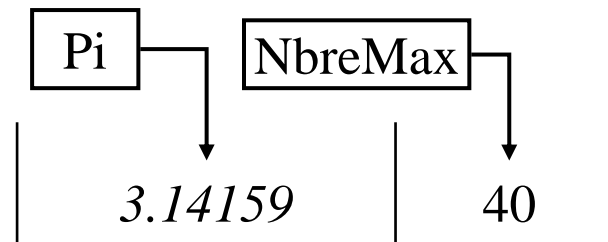
Déclaration de Constantes

```
Const Nom_type nom_const1 = valeur [, nom_const2 = valeur ,...]
```

Exemples

```
Const Réel  Pi  = 3.14159  
Const Entier NbreMax = 40
```

Représentation



Règles de nommage des variables et constantes

Le nom d'une variable ou d'une constante doit :

- Être composé de **lettres**, de **chiffres** et du caractère _
- Commencer obligatoirement par une **lettre** ou par _
- Différent d'un **mot clé**

III. Notion d'action

Opération qui s'applique sur un ou plusieurs objets. Elle peut **définir**, **modifier** ou **restituer** leurs états



Exemples

Action de lecture(appel)	: Lire(n1,n2)
Action calcul (affectation)	: $S \leftarrow P_i * R * R$
Action d'appel	: $M \leftarrow \text{Min}(n1,n2)$
Action d'écriture(appel)	: Ecrire ("Le minimum = " , M)

Remarques

- Une action peut modifier l'état d'un ou plusieurs objets
- Une action peut être **simple** (Appel , affectation) ou **composée** (séquence d'actions)
- L'exécution d'une action peut être contrôlée par une structure :
 - **inconditionnelle** : s'exécute sans aucune condition
 - **conditionnelle** : s'exécute selon la valeur d'une condition
 - **répétée** : s'exécute un certain nombre de fois

Action d'affectation

Consiste à placer dans une variable déclarée une valeur appartenant à son domaine par :

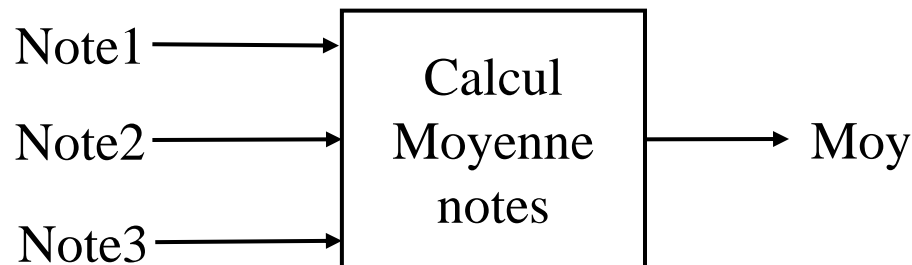
NomVariable \leftarrow expression ;

- Une expression est une suite formée **d'opérateurs** et **d'opérandes**.
- Un opérande est valeur qui peut être représentée par une **constante littérale**, une **variable**, une **constante symbolique** ou un **appel** à une fonction.

Exemples

Corrects	Mauvais
$A \leftarrow 10$	$a + 1 \leftarrow 3$ On n'affecte pas à une expression
$B \leftarrow A$	$A \leftarrow 3B$ 3B n'est ni un identificateur ni une expression
$C \leftarrow 2 * \text{racine}(B) / A$	$\text{Sin}(x) \leftarrow a$ On n'affecte jamais à une fonction
$L \leftarrow (A < B) \text{ ET } (A < C)$	$\text{NomConst} \leftarrow \text{Val}$ On n'affecte jamais à une constante

Ecrire un algorithme qui permet de calculer et d'afficher la moyenne de 3 notes.



Algorithme Moyenne

Réel Note1 , Note2 , Note3, Som, Moy

Début

Lire (Note1,Note2,Note3)

Som \leftarrow Note1

Som \leftarrow Som + Note2

Som \leftarrow Som + Note3

Moy \leftarrow Som / 3

Ecrire ("La moyenne est de : ", Moy)

Fin

Lecture/Ecriture des données

Action de transfert de données de l'extérieur (clavier) vers la RAM pour la lecture et inversement de le RAM vers l'extérieur (écran)

Lire (nom_var1 [,nom_var2,...])

Ecrire (expression1|Littéral [, expression2| Littéral, ...])

Exemple

Calculer le montant d'une facture pour la vente d'un seul produit à une quantité et à un prix donnés

Algorithme **Facture**

Const Réel Ttva = 0.2

Réel PU, MHT, MTTC

Entier Qte

Début

Ecrire("Quantité : ")

Lire (Qte)

Ecrire ("Prix Unitaire : ")

Lire (PU) ;

MHT \leftarrow Qte * PU ;

MTTC \leftarrow MHT * (1 +Ttva) ;

Ecrire ("Montant HT :" ,MHT)

Ecrire ("Montant TTC :" ,MTTC)

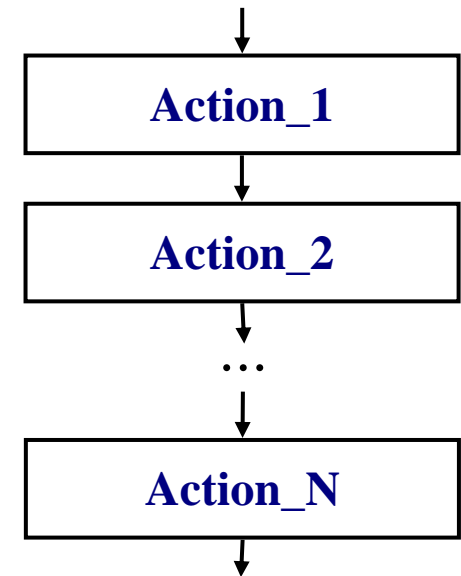
Fin

IV. Les Structures de contrôle

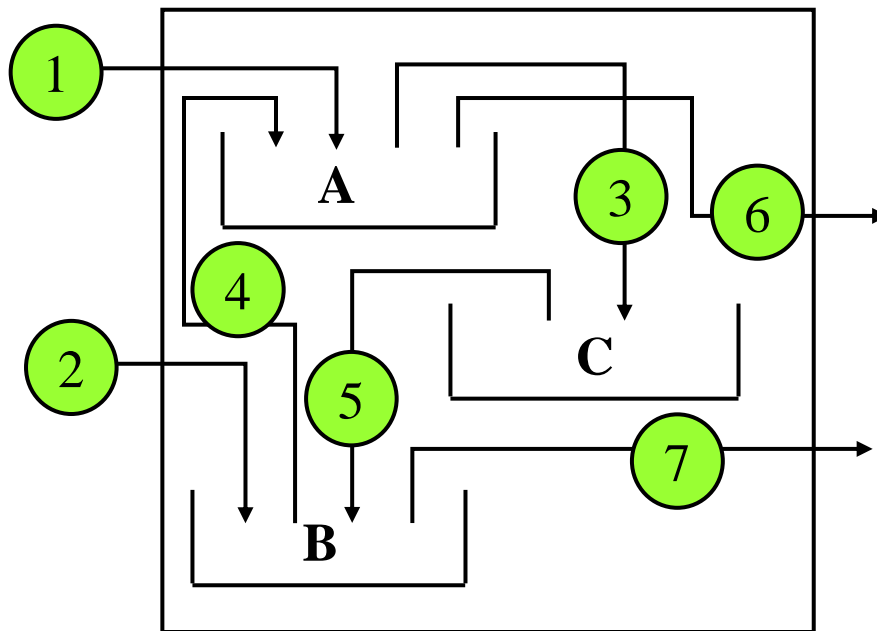
1. Séquence d'actions

suite d'actions simples ou composées.
Elle représente une structure de base
appelée aussi **enchaînement**

Action_1
Action_2
...
Action_N



Exemple : Permutation de deux nombres



Algorithme **Permutation**

Réel A , B , C

Début

Lire (A , B)

C ← A

A ← B

B ← C

Ecrire ("A = " , A , " B = " , B)

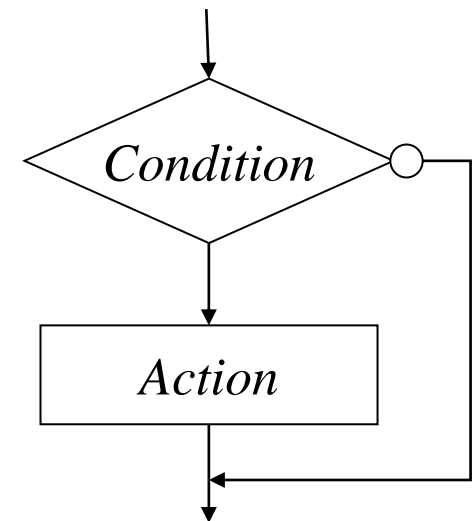
Fin

2. Actions Conditionnelles

Une action conditionnelle est une structure de contrôle qui permet de vérifier une condition et de choisir entre deux chemins possibles

l'alternative simple

```
Si Condition Alors
    Action
Fsi
```



Exemple : Déterminer le signe d'un nombre x .

Algorithme **Signe_Nombre**

```
Entier      x
Caractère   Sgn
Début
    Ecrire ("Donnez x : ")
    Lire (x)
    Sgn ← '+' ;
    Si  $x < 0$  alors
        Sgn ← '-' ;
    Fsi
    Ecrire ("Le signe de ", x, "est : " , Sgn )
Fin
```

l'alternative complète

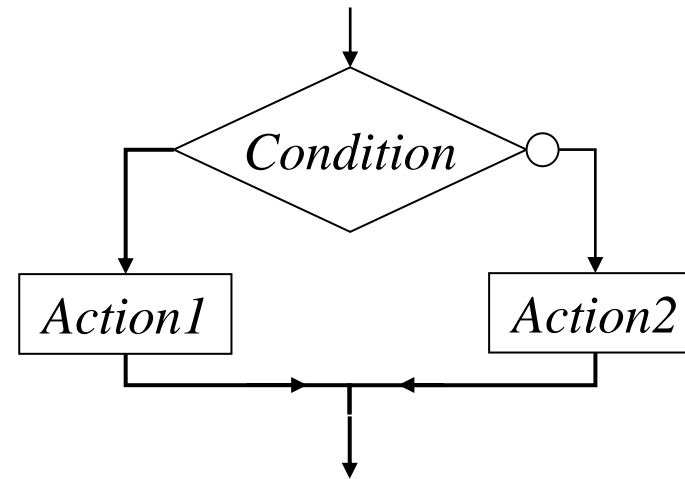
Si *Condition* **Alors**

Action1

Sinon

Action2

Fsi



Exemple : Trier deux nombres réels par ordre croissant

Algorithme **Trie2**

Réel a, b

Début

Ecrire(" a , b :")

Lire (a , b)

Si a < b alors

Ecrire ("a = ", a , " b = " , b)

Sinon

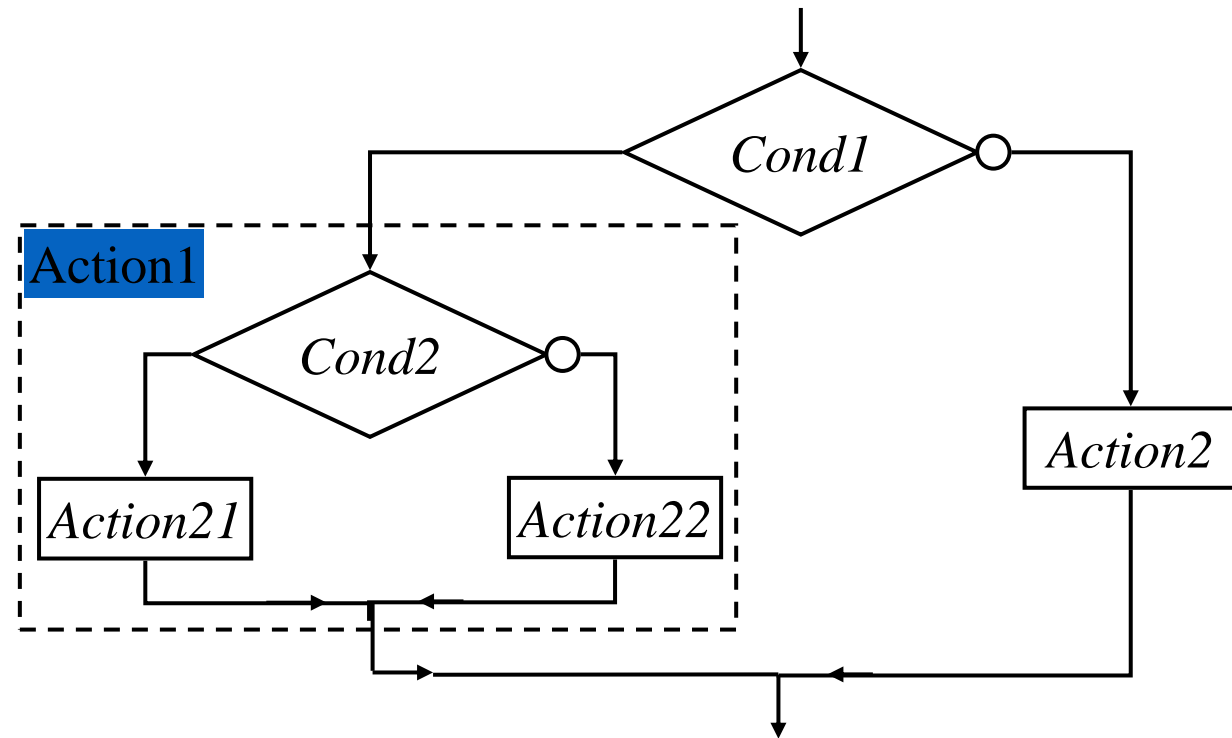
Ecrire ("b = ", b , " a = " , a)

Fsi

Fin

Imbrication de si

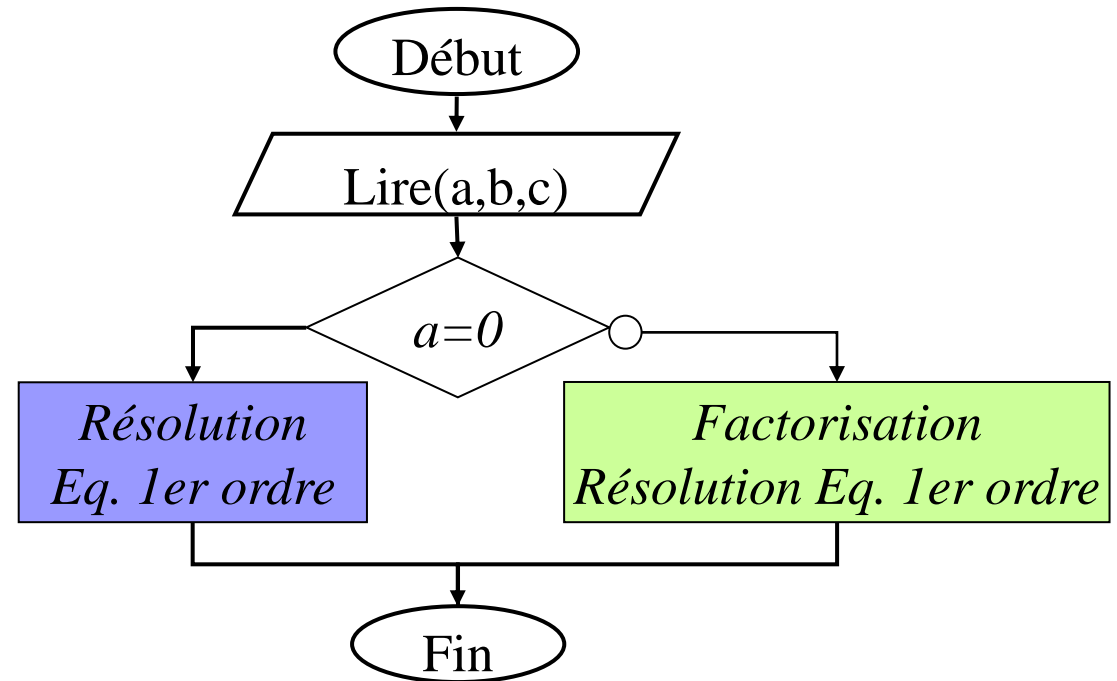
Si *Cond1* **Alors**
 Si *Cond2* **Alors**
 Action21
 Sinon
 Action22
 Fsi
 Sinon
 Action12
 Fsi



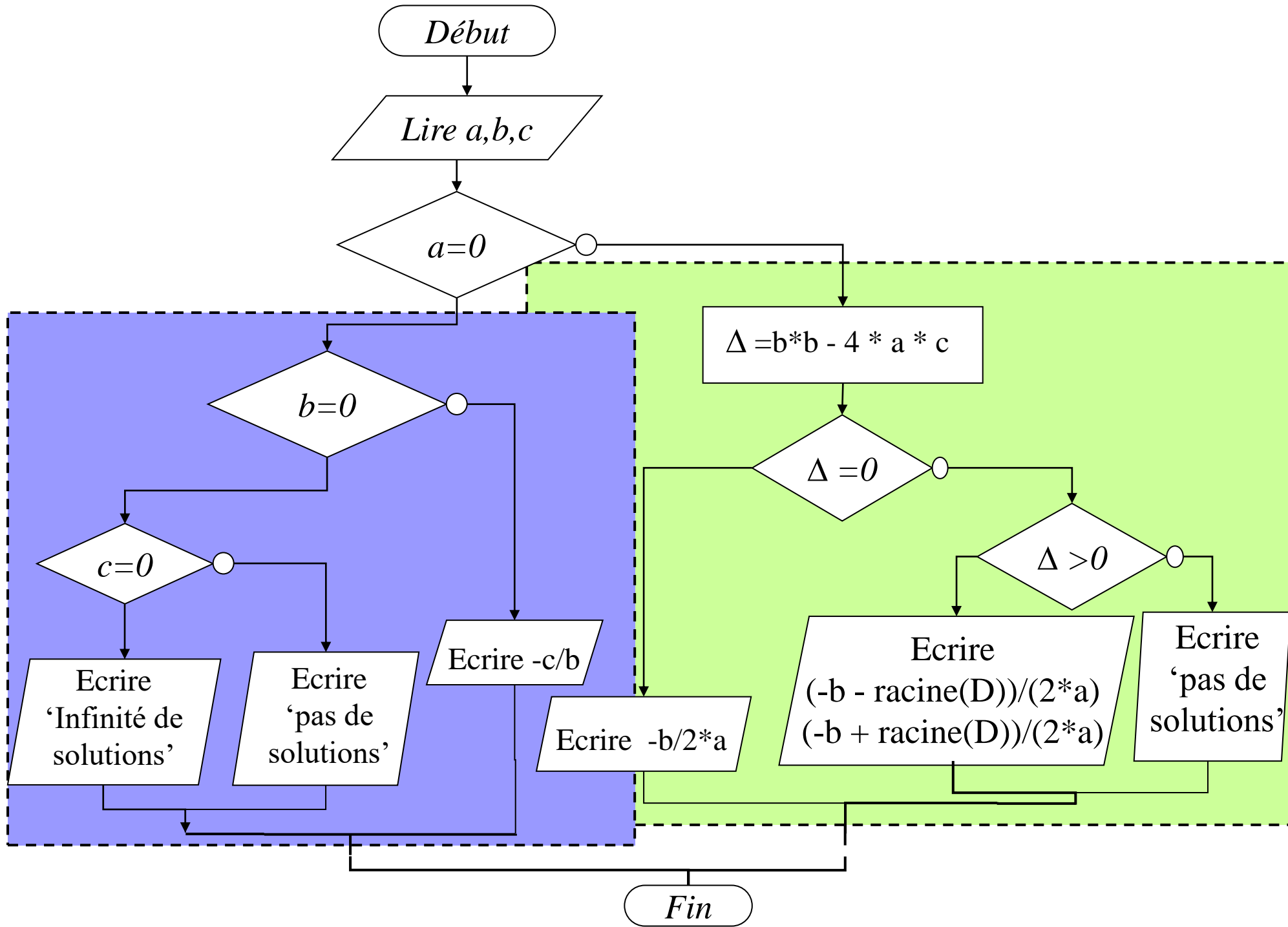
Exemple

Algorithme de résolution
 d'équation de second degré :

$$a x^2 + b x + c = 0$$



Chap II : Les Concepts de base



Algorithme Eq_Second_Degré

Réel a, b, c , Delta

Début

Ecrire ("Donner les coefficients : ")

Lire(a,b,c)

Si a = 0 alors

Si b = 0 alors

Si c = 0 alors

Ecrire ("Infinités de solutions")

Sinon

Ecrire ("Pas de solutions")

FSi

Sinon

Ecrire ("Equation de 1er degré, une racine réelle : ", -c/b) ;

FSi

Sinon

Delta $\leftarrow b*b - 4*a*c$;

Si Delta = 0 Alors

Ecrire ("Une racine réelle double : " , -b/(2*a))

Sinon

Si Delta > 0 Alors

Ecrire("Deux racines réelles : x1 = ", $(-b + \text{racine}(\text{Delta})) / (2*a)$,
" x2 = ", $(-b + \text{racine}(\text{Delta})) / (2*a)$)

Sinon

Ecrire("Pas de solutions réelles ")

FSi

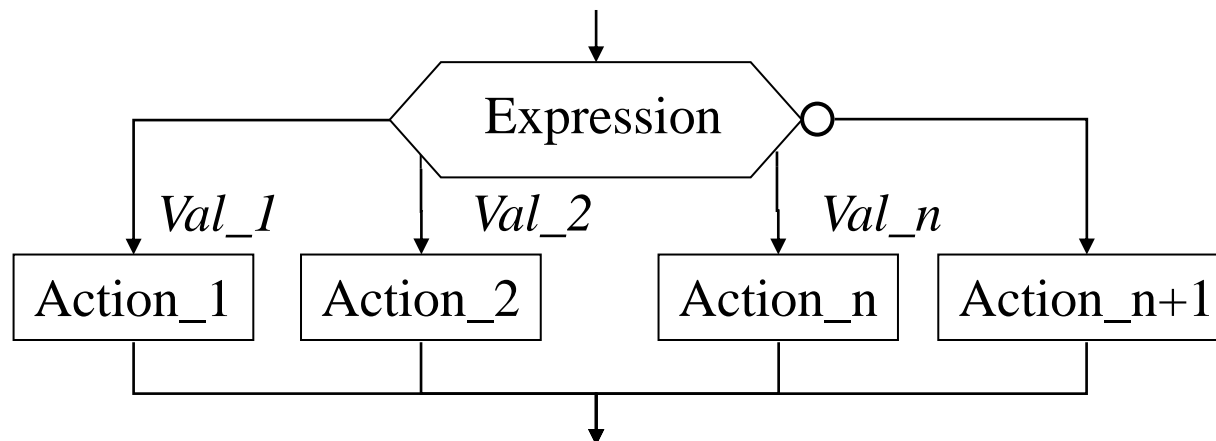
FSi

FSi

Fin

3. Structure sélective

Permet d'effectuer une étude de cas composée de plusieurs choix qui dépendent de la valeur d'une expression appelée **sélecteur**.



Suivant Expression **faire**

Val_1 : Action_1

Val_2 : Action_2

.

.

Val_n : Action_n

Sinon : Action_{n+1}

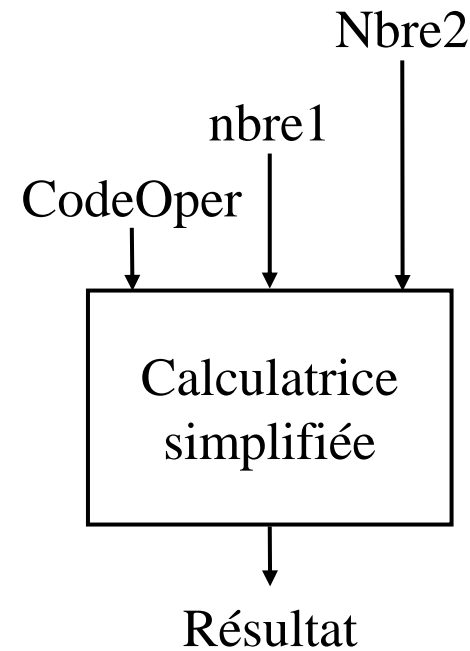
Fsuivant

Chap II : Les Concepts de base

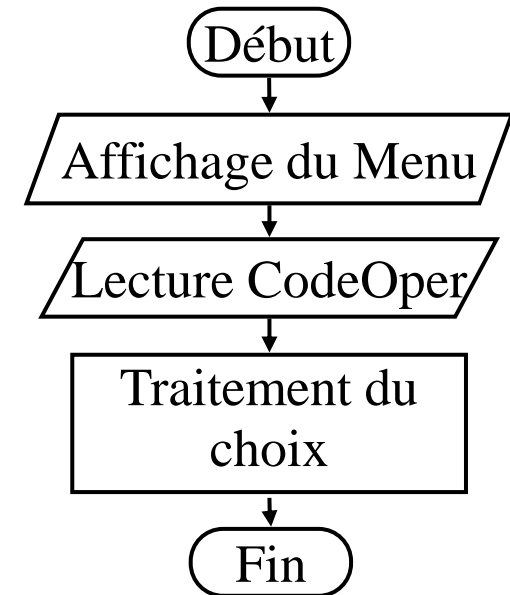
Exemple

Ecrire un algorithme pour réaliser une **calculatrice simplifiée**. Cette calculatrice doit être utilisée par un élève pour effectuer des **opérations arithmétiques** (+, -, *, /, Div, Mod) sur des **entiers**. L'opération à effectuer est choisie à partir d'un **menu** qui présente les différentes options possibles. Un numéro séquentiel est associé à chacune de ces options.

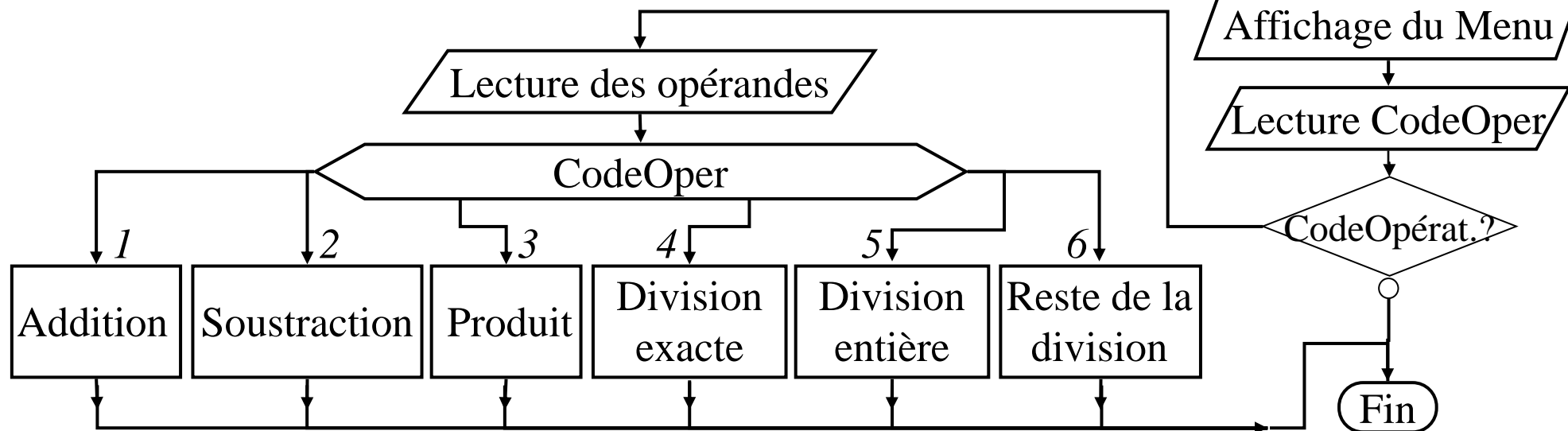
Spécification



Organigramme 1



Organigramme 2



Algorithme Calculette

Entier Nbre1, Nbre2, CodeOper

Début

/ Affichage du Menu */*

Ecrire ("Calculatrice simplifiée : ")

Ecrire ("Addition 1 ")

Ecrire ("Soustraction 2 ")

Ecrire ("Produit 3 ")

Ecrire ("Division exacte 4 ")

Ecrire ("Division entière 5 ")

Ecrire ("Reste de la division 6 ")

/ Lecture de l'opération choisie */*

Ecrire ("Entrez le code de l'opération : ")

Lire (CodeOper)

/ Traitement du choix */*

Si (Codeoper \geq 1) ET (CodeOper \leq 6) **alors**

Ecrire ("Entrez les valeurs des opérandes : ")

Lire(Nbre1,Nbre2)

Suivant CodeOper **Faire**

Cas 1 :

Ecrire ("Le résultat est : " , Nbre1 + Nbre2);

Cas 2 :

Ecrire ("Le résultat est : " , Nbre1 - Nbre2)

Cas 3 :

Ecrire ("Le résultat est : " , Nbre1 * Nbre2)

Cas 4 :

Si Nbre2 \neq 0 **Alors**

Ecrire ("Le résultat est : " , Nbre1 / Nbre2)

Sinon

Ecrire ("Division par 0 impossible")

FSi

Cas 5 :

Si Nbre2 \neq 0 **Alors**

Ecrire ("Le résultat est : " , Nbre1 Div Nbre2)

Sinon

Ecrire ("Division par 0 impossible")

FSi

Cas 6 :

Si Nbre2 <> 0 **Alors**

Ecrire ("Le résultat est : " , Nbre1 Mod Nbre2)

Sinon

Ecrire ("Division par 0 impossible")

FSi

FSuivant

FSi

Fin

4. Structures répétitives (boucles)

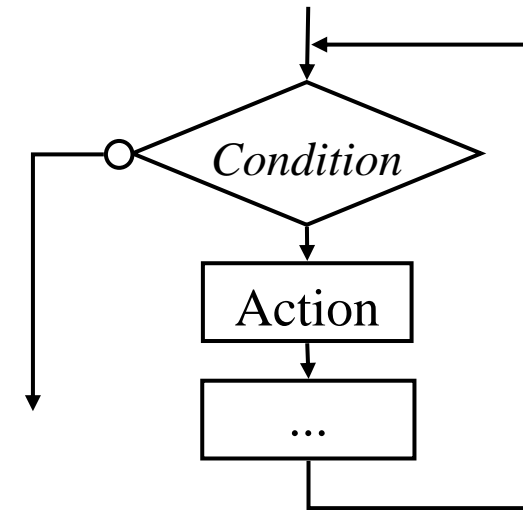
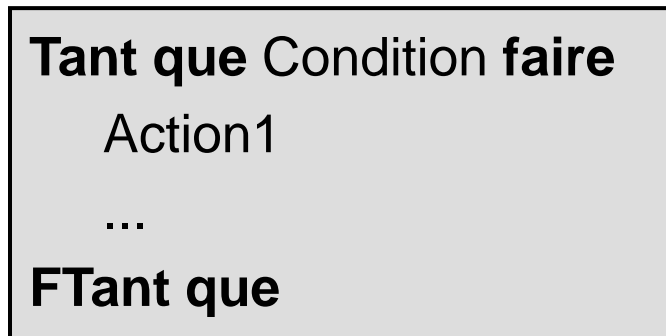
Une structure répétitive consiste à répéter plusieurs fois l'exécution d'une ou de plusieurs actions.

On distingue deux types actions répétitives :

- Action répétitive dont le nombre d'itérations est à priori inconnu. Dans ce cas c'est l'exécution de l'action itérative qui déterminera son arrêt : **Tant que , Répéter**
- Action répétitive dont le nombre d'itérations est connu a priori : **Pour**

La boucle Tant que

Tant qu'une condition est vérifiée, On répète une ou plusieurs actions



Remarques

- si la condition n'est pas vérifiée au départ, on n'exécutera jamais la boucle.
- Dans le corps de la boucle, l'état de la condition doit être modifié de manière à éviter une boucle infinie.

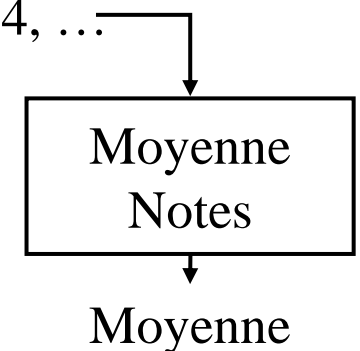
Exemple

Calculer la **moyenne** des notes d'une liste de copies.

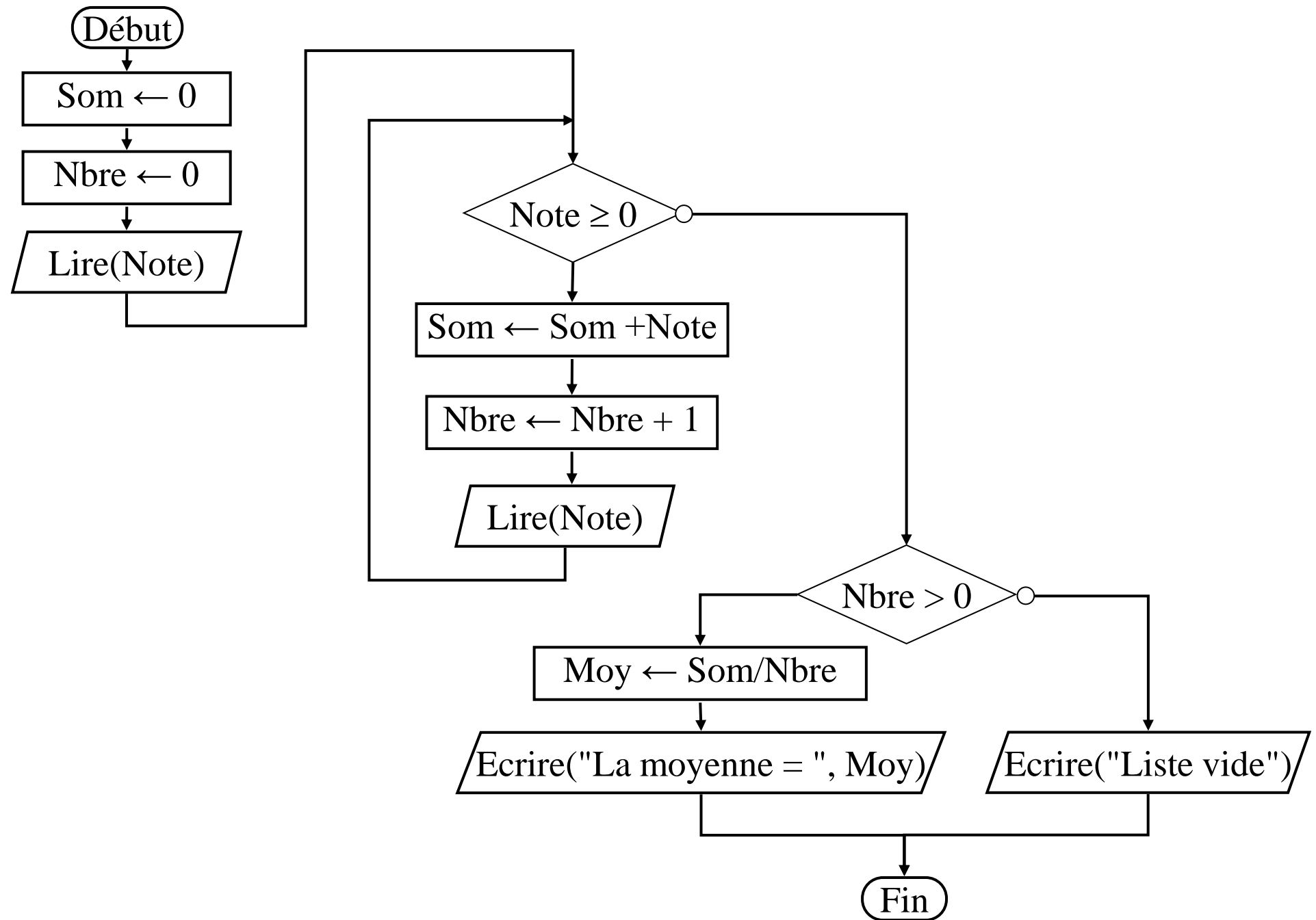
Le **nombre** de copies est a priori **inconnu**. La fin de la liste des copies est indiquée par l'ajout d'une copie **fictive** contenant une note **négative**. On suppose pour cela que les notes sont comprises entre 0 et 20.

Spécification

Notes : -1, 14, ...



Organigramme



Algorithme **Moyenne_Notes**

Réel Note, Moy, Som

Entier Nbre

Début

*/*initialisation*/*

Som \leftarrow 0;

Nbre \leftarrow 0 ;

*/*Saisie des notes et calcul de la somme des notes */*

Ecrire("Entrez une note : ")

Lire(Note)

/ Calcul de la somme et comptage */*

Tant que Note \geq 0 **faire**

Som \leftarrow Som + Note ;

Nbre \leftarrow Nbre +1 ;

Ecrire("Entrez une note : ")

Lire(Note)

Ftant que

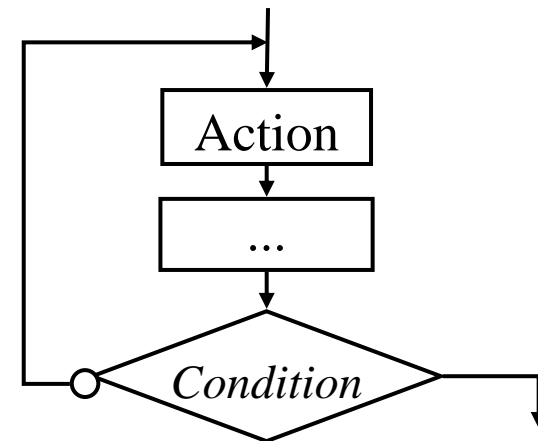
*/*Cacclul et affichage de la moyenne de la moyenne */*

```
Si Nbre>0 alors
    Moy ← Som / nbre ;
    Ecrire ("La moyenne des ", Nbre, " notes est de : ", Moy )
Sinon
    Ecrire("Liste de notes vide ")
FSi
Fin
```

La boucle Répéter .. jusqu'à

Consiste à répéter une ou plusieurs actions jusqu'à ce qu'une condition soit vérifiée.

```
Répéter
    Action1
    ...
Jusqu'à Condition
```



Remarques

- La boucle est exécutée au moins une fois.
- Dans le corps de la boucle, l'état de la condition doit être modifié de manière à éviter une boucle infinie.

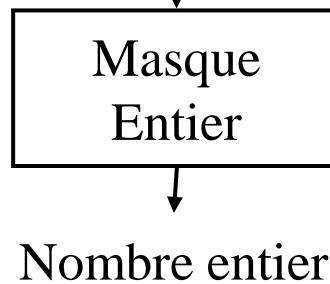
Exemple : Masque de saisie d'un entier

Ecrire un algorithme qui attend une suite de 5 chiffres représentant un nombre entier. L'algorithme acceptera tout caractère frappé mais :

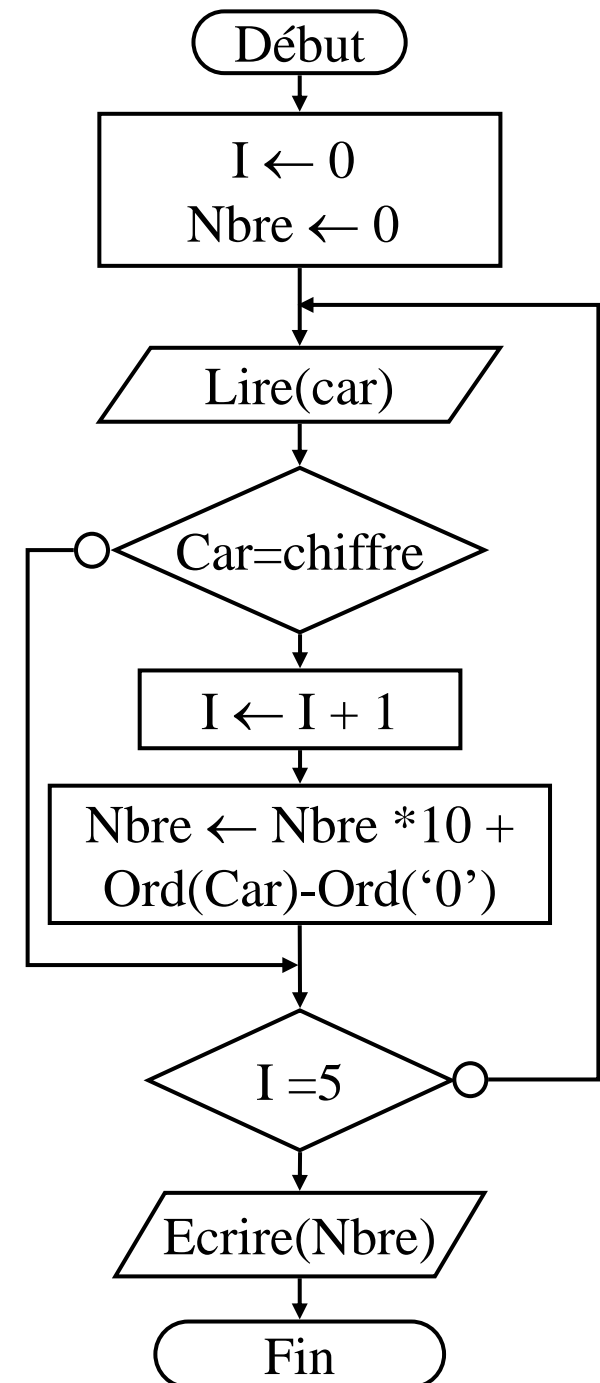
- il n'affichera que les chiffres
- il ne tiendra pas en compte des caractères différents d'un chiffre
- il attend d'avoir reçu effectivement 5 chiffres

Spécification

Caractères : C,1,e,...



Organigramme



Algorithme **Masque_Entier**

const Entier NbCh = 5

Entier I , Nbre

Caractère C

Début

/ Initialisation */*

I ← 0 ;

Répéter

Lire(C) ;

Si C >='0' ET C <='9' Alors

Nbre ← Nbre *10 +Ord(Car)-Ord('0') ;

I ← I+1;

fsi

Jusqu'à I = Nbch ;

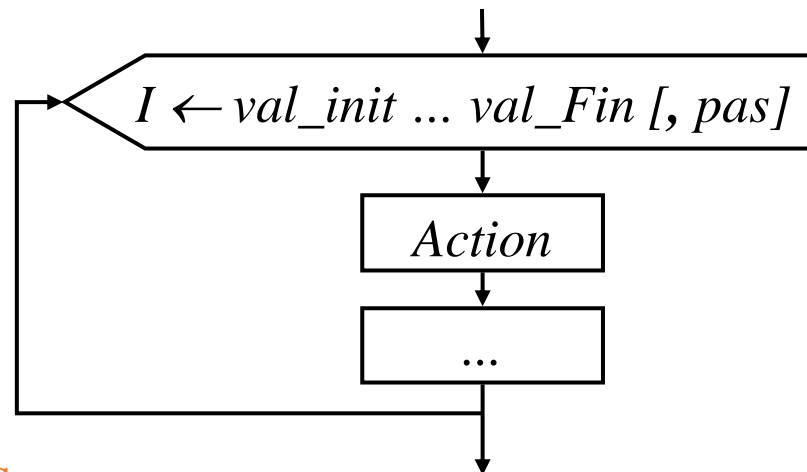
Ecrire("Le nombre saisi est : " , Nbre)

Fin

Chap II : Les Concepts de base

La boucle Pour On répète une ou plusieurs actions un nombre de fois connu à priori.

Pour $I \leftarrow \text{val_initiale}$ à val_finale [, Pas] **Faire**
Action
...
FPour



Remarques

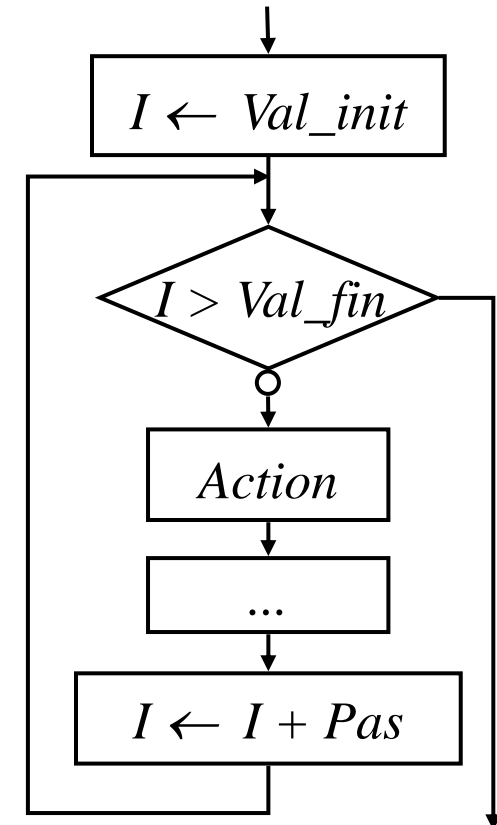
- Si $\text{Val_init} > \text{Val_Fin}$, la boucle ne s'exécutera aucune fois
- Si elle est omise, la valeur du pas est par défaut égale à 1

Exemple Calculer la somme des N premiers nombres premiers.

Spécification

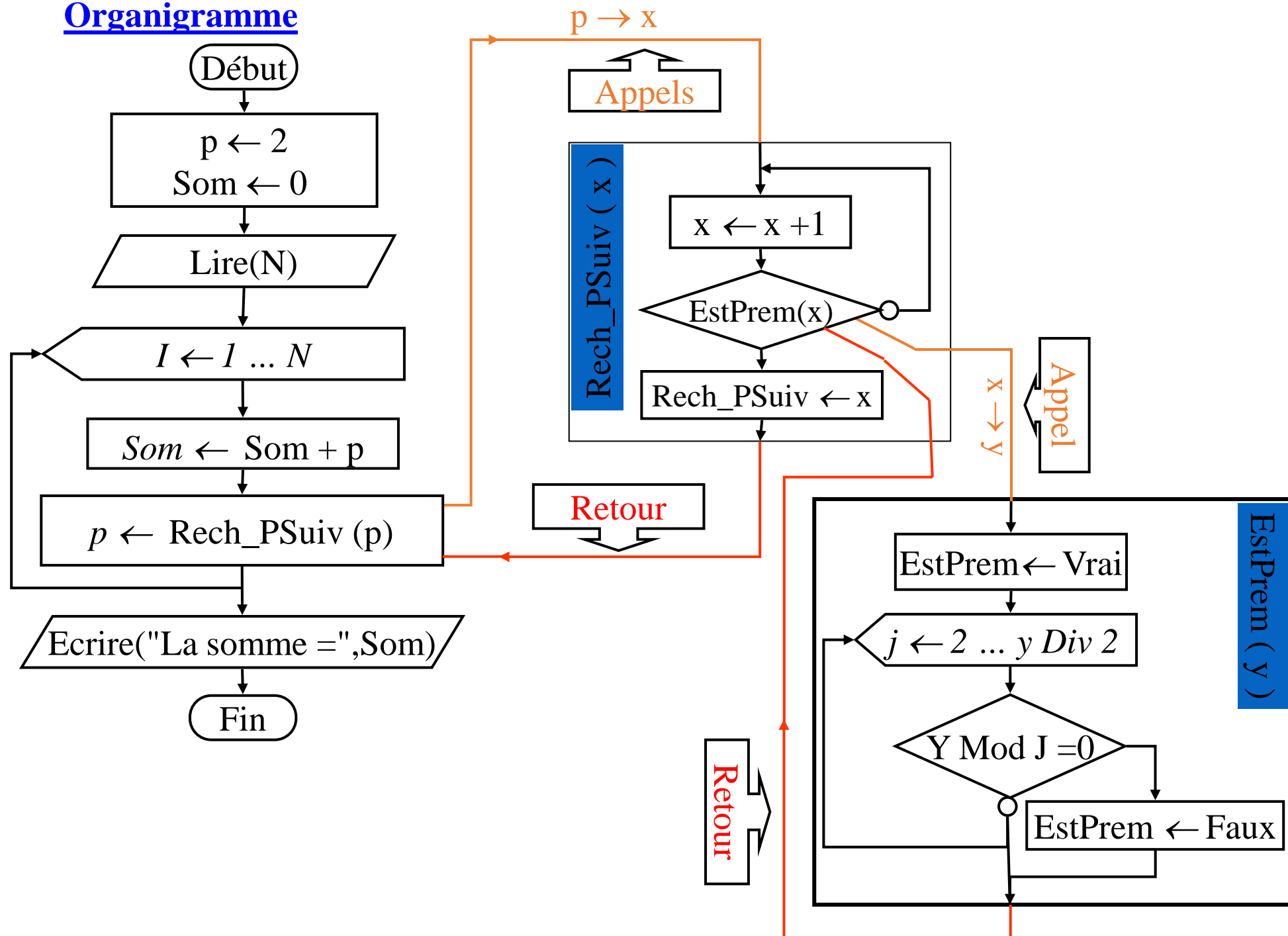


Fonctionnement



Chap II : Les Concepts de base

Organigramme



Algorithme **SomNbrePremiers**

Entier I , N , p , j , Som

Logique EstPrem

Début

p ← 2 ; Som ← 0 /* Initialisation */

Lire(N)

Pour I ← 1 à N **Faire**

 Som ← Som + p ;

 Répéter /* Rechercher le prochain nombre premier */

 EstPrem ← Vrai

 p ← p + 1 /*à partir de 3 on peut incrémenter p de 2

Pour(j ← 2 ; j ≤ p Div 2 ; J ← J+1) /* Examiner si p est premier */

 si p mod j = 0 alors

 EstPrem ← Faux ;

 J ← p ; -- Sortir de la boucle Pour --

 FSi

FPour

 Jusqu'à EstPrem ;

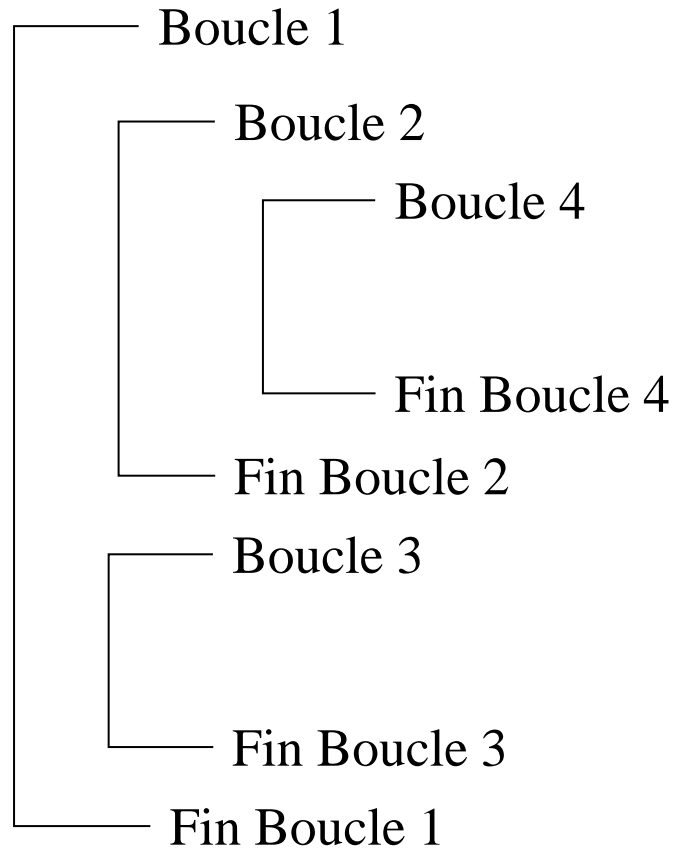
FPour

 Ecrire ("La somme des ", N, "premiers nombre Premiers est : " , Som)

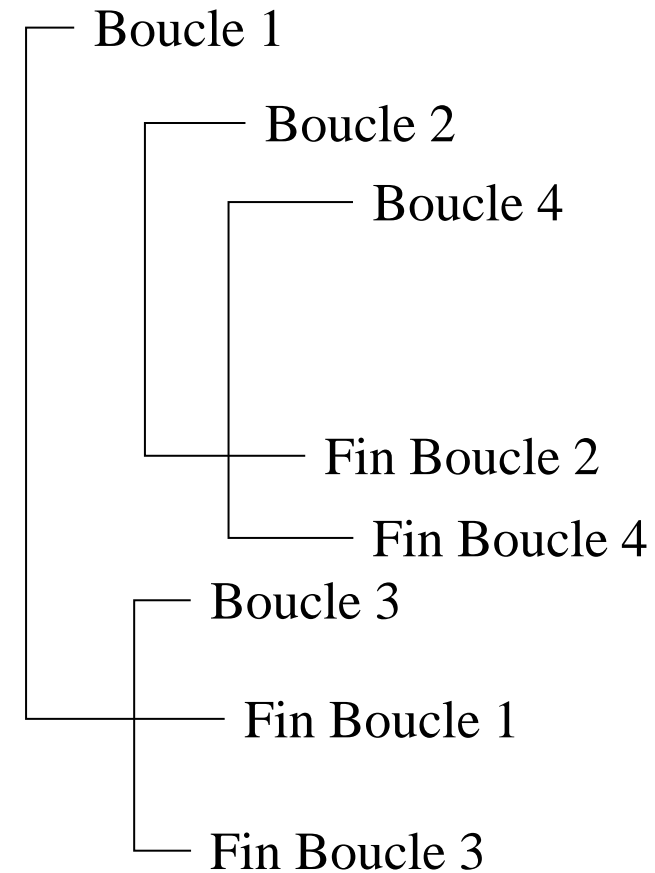
Fin

Imbrications de boucles

Imbrications autorisées



Imbrications interdites



Exemple1 :codage en binaire

Ecrire un algorithme qui reçoit un entier

Naturel et affiche son code binaire minimal

- Diviser le nombre par 2
- Afficher le reste de la division
- Répéter jusqu'à obtenir un quotient nul

Spécification

