

Ch. 1 Introduction

Objectifs

Définir les concepts

- Ordinateur
- Programme
- Langage de programmation
- Information
- Traitement de l'information

▪ Notion d'ordinateur

Machine électronique ultra rapide possédant :

- Unité centrale de traitement(UCT, CPU, Processeurs),
- Mémoire pour stocker les données à traiter (RAM),
- Des unités d'entrées et de sorties (E/S) (ports) pour communiquer avec l'extérieur : périphériques.

- Extérieur comprend (périphériques)
 - Clavier, scanner
 - Les supports de stockage(disque dur, disque optique...)
 - Imprimante
 - Souris

- Notion de programme
 - Un ordinateur est une machine programmable.
- ⇒ besoin de programmeurs pour programmer ou écrire des programmes.

- Notion de programme

Programme est une

suite

finie d'instructions

élémentaires exécutables par ordinateur.

- Langage de programmation

L'ordinateur doit exécuter des instructions d'un programme.

=> Ces instructions doivent être comprises ou acceptées par l'ordinateur

=> Instructions doivent être écrites dans un langage compris par l'ordinateur

=> Ce langage s'appelle langage de programmation: C, Pascal, VB, JAVA

- Notion d'information

Un ordinateur est une machine de traitement d'information.

- Information(suite)

- Une information est un **renseignement** qui porte sur un objet (nom d'un étudiant, intitulé d'un module, rayon d'un cercle...).
- Une information est un **critère qui réduit le domaine** où on cherche la réponse une question (réduit l'incertitude).

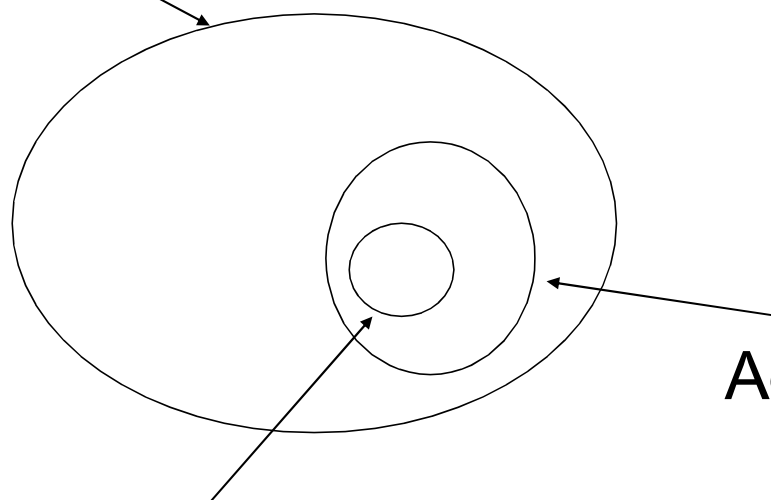
■ Exemple

Question : Chercher le plus grand mot d'une langue donnée. Soit "L" le nom de cette langue.

Considérons les Informations suivantes

- +Ce mot est un adverbe.
- +Ce mot commence par la lettre X
- Ce mot dépasse 2 caractères.

Les mots de la langue L



Adverbes

Adverbes commençant par X

- Donnée et information

Donnée=information

mais

Données =aspect physique de l'information.

ou

Information =données avec un sens.

- Traitement de l'information

La tâche principale d'un ordinateur est le traitement de l'information. Ce traitement se compose de 4 fonctions :

- Saisie des données (entrées): clavier...
- Mémorisation des données: stockage
- Opérations sur les données: calcul, tri...
- Restitution des résultats: affichage, impression, fichier...

Module : Infor3

Initiation à la programmation

Ch. 2: Algorithmique

Ch. 2: Algorithmique

Objectifs

Maîtriser les concepts ou les notions :

- Instruction
- Algorithme
- Donnée
- Variable
- Constante

- Notion d'instruction

Une instruction est un ordre qu'on demande à un ordinateur d'exécuter.

L'exécution d'une instruction porte souvent sur des données.

- Exemple d'instruction

Calculer $2 + 3$ est une instruction ,
+ est le nom de l'opération (**opérateur**),
2 et 3 sont les données (**opérandes**).

- Exemple d'instruction

Lire une valeur au clavier : instruction de lecture. On la représente par le mot :

Lire

- Exemple d'instruction

Écrire une valeur à l'écran :
Instruction d'écriture. On la représente par le mot

Ecrire

- Étapes de résolution d'un problème

Ordinateur = machine programmable

↓
Rédaction de programme

↓
Rédaction d'algorithme

↓
Etapes à suivre pour rédiger un algorithme?

- Les étapes de résolution d'un problème en programmation
 - i. Établir la liste des **données en entrée**(données à saisir), la liste des **données en sortie**(résultats : données à afficher) et les liens entre elles
 - ii. Construire un chemin de résolution qui permet d'obtenir les données en sortie à partir des données en entrée. C'est ce qu'on appelle un **schéma de résolution**.
 - iii. Décrire le schéma de résolution en termes d'instructions élémentaires acceptées par ordinateur. C'est l'algorithme.

■ Exemple

Problème : Automatiser le calcul de la surface d'un disque

i. Identification des données d'entrées et de sorties

- Donnée en entrées : rayon, pi
- Données en sorties : surface
- Relations entre les données : $\text{surface} = \text{pi} \times \text{rayon} \times \text{rayon}$.

ii. Chemin de résolution

- Donner une valeur à rayon (affectation ou une lecture)
- Calculer $\text{pi} \times \text{rayon} \times \text{rayon}$
- Mettre $\text{pi} \times \text{rayon} \times \text{rayon}$ dans surface (affectation)
- Afficher la valeur de surface (Ecrire).

iii. Traduire le chemin en algorithme

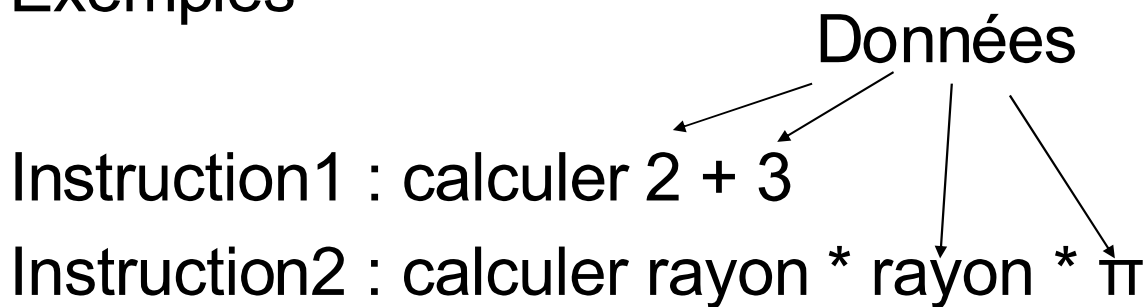
■ Notion d'algorithme

Un algorithme est une *suite finie* d'instructions élémentaires exécutables par ordinateur.

- Notion de donnée

Les données sont les objets manipulés par les instructions d'un algorithme.

- Exemples



- Nature des données

Les données peuvent être

- Données variables ou simplement variables

Exemple : rayon (calculer la surface de (+) disque)

- Nature des données(suite)

Les données peuvent être

- Données constantes ou simplement constantes

Exemple π , ou la valeur 3.14

- π est dite constante symbolique,
- 3.14 dite constante littérale.

- Autres exemples de données constantes

- La constante de coulomb C sa valeur est $8.98 * 10^9$

- La charge $E=1.6*10^{-19}$

Remarque

Une constante symbolique peut être manipulée directement par son nom ou par sa valeur littérale.

Exemple

- $\text{rayon} * \text{rayon} * \pi$

ou

- $\text{rayon} * \text{rayon} * 3.14$

- Déclaration des données

Les variables et les constantes symboliques utilisées dans un algorithme doivent être déclarées.

Cette déclaration inclut le **nom**, le **type** et la **nature** de la donnée.

■ Le nom

Le nom permet de distinguer la donnée parmi les autres données de l'algorithme. Ce nom doit être un identificateur.

Identificateur : nom qui commence par une **lettre** ou le souligné suivi de lettres ou de chiffres ou le caractère souligné.

On préfère que ce nom soit significatif.

■ Exemples

R, S, pi, P, V, T, Adresse, Ville.

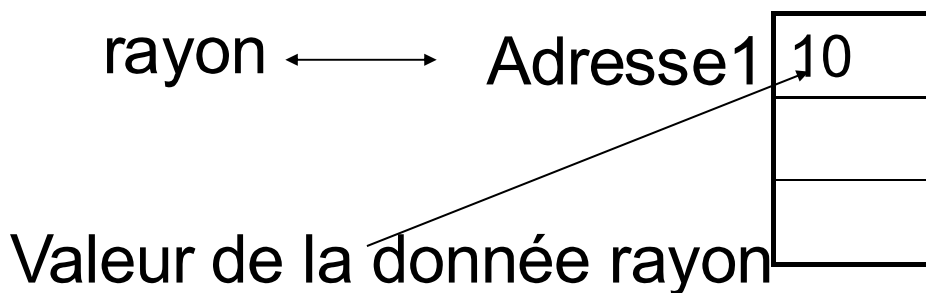
■ Contre-exemples

1nom,
nom d'étudiant,
nom!

- Lien entre nom de données et mémoire

A chaque nom de donnée déclarée est associée une adresse physique d'une case mémoire de l'ordinateur.

Cette case mémoire contient la valeur de la donnée.



- Le type

Le type désigne l'ensemble ou l'intervalle des valeurs que peut prendre la donnée.

On s'intéresse aux types simples suivants:

- Entiers(1,-1...),
- Réels(2.3...),
- Caractères('a','!'),
- Chaîne de caractères (suite de caractères entre guillemets, " bonjour").

■ La nature

La nature d'une donnée indique si la donnée est constante ou variable.

- La donnée constante ne change pas de valeur dans l'algorithme.
- La donnée variable peut changer de valeur dans l'algorithme.

■ Syntaxe de déclaration d'une variable

Les variables se déclarent dans une rubrique **Variables** avec la syntaxe de déclaration suivante

nom_variable : type;

Exemple

Variables

Rayon : réel;

Surface : réel ; n : entier

ou

Variables

rayon, surface : réel ;

n : entier ;

Rubrique variable

Déclaration des variables

- Syntaxe de déclaration d'une constante

Les constantes se déclarent dans une rubrique **Constantes** avec la syntaxe suivante

nom_constante_symb=valeur_constante_litt;

Exemple

Constantes

pi =3.14 ;

taux=2 ;

- Règle à respecter

Les variables se déclarent après les constantes.

Exemple

Constantes

pi=3.14 ;

taux=2 ;

Variables

Rayon, surface : réels;

■ Remarques

- Les constantes littérales de type caractères sont entre apostrophes.
- Les constantes littérales de type chaîne de caractères sont entre guillemets.

■ Exemples

Constantes

```
GENRE1='F';  
GENRE2='M';  
SALUT="salam";
```

■ Notion expression

Une expression est une combinaison logique

- d'identificateurs,
- de valeurs ,
- d'opérateurs (+, *, /, -, %...) et
- d'autres symboles tels que (), .

Exemples

- 1
- $2+3$
- rayon * rayon * pi
- $(\text{Pi} \cdot \text{rayon})^2$

Contre-exemples

2^*+3
 $(2+5^*2$

■ Exemples

- 1
- $2+3$
- rayon * rayon * pi
- $(\text{Pi} \cdot \text{rayon})^2$

■ Contre-exemples

- 2^*+3
- $(2+5^*2$

- Instructions élémentaires :

Instruction d'affectation

sert à affecter la valeur d'une expression à une variable. On la note par le symbole



Pour affecter la valeur d'une expression à une variable on écrit :

Nom_variable ← expression;

- Exemple1

rayon ← 5;

Après cette instruction la valeur de rayon est 5.

La **case mémoire** associée à la variable rayon contient la valeur 5.

Avant affectation

rayon

?

Après affectation

rayon

5

- Exemple 2

rayon ← 5;

surface ← pi*rayon*rayon;

Après ces instructions la valeur de surface est 78.5.

Avant affectation

rayon

surface

?
?

Après affectation

rayon

surface

5
78.5

- Remarque

La partie gauche de l'affectation doit être une **variable**.

- Contre-exemples

1 ← 5;
surface*2 ← pi*rayon*rayon;

- Instruction de lecture

Permet de **lire** des valeurs à partir du **clavier** et les **affecte** aux **variables**.

La syntaxe de cette instruction est :

Lire (var1, var2, ...);
ou bien
lire (var1, var2, ...);

■ Exemple 1

Lire(rayon);

A l'exécution de cette instruction, quand on saisit la valeur 8 au clavier elle sera la valeur de la variable rayon.

Avant lecture

rayon	<input data-bbox="866 730 1265 792" type="text" value="?"/>
-------	---

Après lecture

rayon	<input data-bbox="866 880 1265 943" type="text" value="8"/>
-------	---

■ Exemple 2 : lire les valeurs de plusieurs variables

Variables

nom : chaîne;

age : réel;

...

Lire(nom, age);

nom et age sont des variables.

- Remarque

Les arguments de Lire doit être des variables.

- Contre-exemples

lire(3);

lire(x+y);

■ Instruction d'écriture

L'instruction d'écriture (écriture à l'écran) permet **d'afficher à l'écran** les valeurs des **variables** ou

expressions

après les avoir évaluées.

Sa syntaxe est la suivante:

**Ecrire (expr1, expr2...); ou bien
écrire(expr1, expr2...);**

■ Exemples

- Ecrire (rayon); affiche la valeur de rayon :5
- Ecrire (surface); affiche à l'écran la valeur de surface :78.5
- Ecrire (pi*rayon*rayon); affiche aussi 78.5
- Ecrire("surface"); affiche le mot surface.

▪ Structure générale d'un algorithme

Algorithme Nom_algorithme;

 Constantes

 Liste_de_constants;

 Variables

 Liste_de_varaibles;

 Début

 Liste_instructions;

 Fin.

▪ Exemples d'algorithmes

- Calcul de la surface d'un disque
- Calculer de la somme des n premiers entiers
 $1+2+3+\dots+n$
- Permutation des valeurs de deux variables.

■ Exemple 1 surface d'un disque

i. Identification des données d'entrées et de sorties

- Donnée en entrées : rayon, π
- Données en sorties : surface
- Relations entre les données : $\text{surface} = \pi * \text{rayon} * \text{rayon}$.

ii. Chemin de résolution

- Donner une valeur à rayon (affectation ou une lecture)
- Calculer $\pi * \text{rayon} * \text{rayon}$
- Mettre $\pi * \text{rayon} * \text{rayon}$ dans surface (affectation)
- Afficher la valeur de surface (Ecrire).

iii. Traduire le chemin en algorithme

■ Exemple 1 (suite)

Algorithme surfaceDisque;

Constantes

$\pi = 3.14$;

Variables

rayon, surface: réels ;

Début

Rayon $\leftarrow 5$;

Surface $\leftarrow \text{rayon} * \text{rayon} * \pi$;

Ecrire("surface=", surface);

Fin.

▪ Exemple 2 : calcul de la somme

somme=1+2+3...n, n est un entier à saisir au clavier.

i. Identification des données d'entrées et de sorties

- Donnée en entrées : n
- Données en sorties : somme
- Relations entre les données : somme= $n*(n+1)/2$

ii. Chemin de résolution

- Donner une valeur à n (lire(n))
- Calculer $n*(n+1)/2$
- Mettre $n*(n+1)/2$ dans somme(affectation)
- Afficher la valeur de somme (Ecrire).

iii. Traduire le chemin en algorithme

▪ Exemple 2 (suite)

Algorithme sommeN ;

Constantes

~~Variables~~

n, somme: entiers ;

Début

Lire(n) ;

somme $\leftarrow n*(n+1)/2$;

Ecrire (somme); ou bien Ecrire("la sommes est:" ,
somme) ;

Fin.

▪ Exemple 3

Permuter les valeurs de deux variables x et y.

Par exemple

Au début x contient 12, y contient 13

Après permutation x contiendra 13 et y contiendra 12.

x 12 y 13

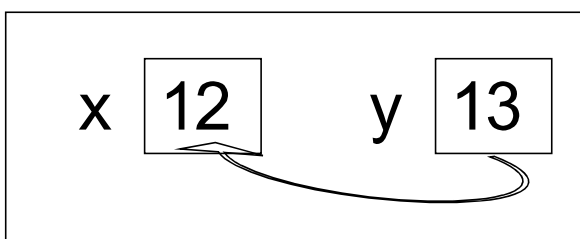
x 13 y 12

▪ Exemple 3 (suite)

Solutions pour la permutation

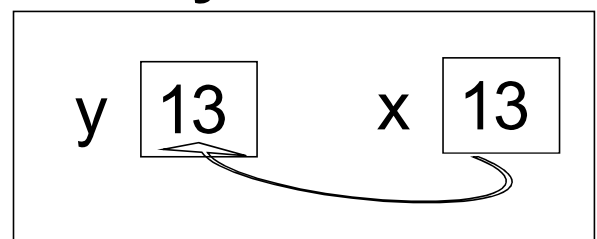
Si on fait $x \leftarrow y$; et $y \leftarrow x$; on aura

$x \leftarrow y$



et

$y \leftarrow x$



→ x et y ont même valeur celui de y (13).

- Exemple 3 (suite)

La raison?

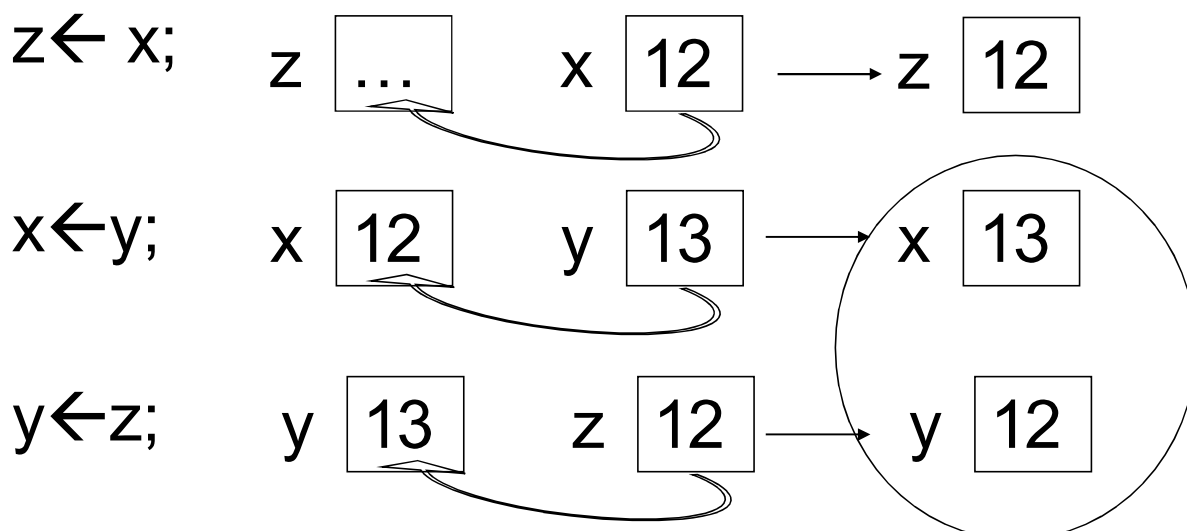
La valeur de x n'a pas été conservée. Elle a été écrasée par la valeur de y(13).

Solution ?

➔ Ajouter une autre variable z pour conserver provisoirement la valeur de x.

- Exemple 3(suite)

Solution



■ Exemple 3 (suite)

i. Identification des données d'entrées et de sorties

- Donnée en entrées : x et y
- Données en sorties : x et y échangées

ii. Chemin de résolution

- Donner une valeur à x;
- Donner une valeur à y;
- Échanger les valeurs de x et y;
- Afficher x et y.

iii. Traduire le chemin en algorithme

■ Exemple 3 (Fin)

Algorithme permutation;

Variables

x,y,z: entiers ;

Début

Lire(x) ;

Lire(y) ;

$z \leftarrow x$;

$x \leftarrow y$;

$y \leftarrow z$;

Ecrire ("la valeur de x après permutation : ", x) ;

Ecrire ("la valeur de y après permutation : ", y) ;

Fin.

- Une autre solution

Pour permuter les valeurs de deux variables : méthode de différences

$x(12)$ et $y(13)$

$x \leftarrow x - y$; $x(12-13)$

$y \leftarrow x + y$; $y(-1+13)$, $x - y + y$

$x \leftarrow y - x$; $x(12-(-1))$, $x + y - x$

- Exercice

Écrire un algorithme qui lit un réel et affiche son carré.

Algorithme carré ;

Variables x, y : réels;

Début

Lire(x);

$y \leftarrow x * x$;

Ecrire ("le carré de ", x , " est : " , y);

Fin.

- Bloc d'instruction

Un bloc d'instructions est une suite d'instructions délimitées par les mots début et fin

Exemple

Début

```
Lire(rayon);  
surface=rayon*rayon*PI;  
Ecrire (surface);
```

Fin.

← Bloc d'instructions

Note. En langage c le mot début est représenté par { et le fin par }.

- Environnement d'un algorithme

L'environnement d'un algorithme est l'ensemble des données déclarées dans les rubriques **constantes** et **variables**

Exemples

- L'environnement de l'algorithme surfaceDisque est constitué de la constante PI, les variables rayon et surface.
- L'environnement de permutation est x, y et z.

■ Les structures de contrôles

Une structure de contrôle sert à contrôler l'exécution d'une instruction ou d'un bloc d'instructions.

- Deux types de structures de contrôles:
 - Structure conditionnelle si l'exécution de l'instruction ou du bloc dépend d'une **condition**
 - Structure répétitive(itérative ou boucle) si l'exécution de l'instruction ou du bloc peut être répétée plusieurs fois(basée aussi sur une condition).

Exemples

- Si **x est positif** alors
Calculer la racine carrée.
- Si **a est différent de 0** alors
Calculer $-b/a$
- Pour tout **i** de 1 jusqu'à 100 faire
Ajouter à s la somme $s+i$.

Contrôle ou condition

Instruction

■ Notion de condition

Une condition est une expression dont la valeur est

soit vraie,

soit fausse

(expression booléenne ou expression de **type booléen**).

■ Exemples et contre-exemples

- $1+2$
- $1.3+5$
- "bonjour"
- 'a'
- $2 > 3$
- $2 < 3$
- $2=3$
- $2 \geq 3$
- $2 \neq 3$

- Remarques
 - Le type booléen={vraie, fausse}
 - On peut aussi déclarer des variables de type booléen.

- Exemple

Variables

test : booléen;

....

test ← 2>3;

- Condition et opérateurs

Les conditions sont exprimées par des opérateurs de comparaison et des opérateurs booléennes :

- Les opérateurs de comparaison sont =, <, <=, >, >= et ≠
- Les opérateurs booléennes s'utilisent avec des opérandes booléens
Et(And), ou (or) et non(not)

- Définition de l'opérateur ET

Soit A et B deux expressions booléennes

L'expression A ET B est vraie ssi

A est vraie et B est vraie.

- Exemples

Variables

test: booléen;

...

- test \leftarrow (2>3) ET (2=2) ;
- test \leftarrow (2<3) ET (2=2) ;
- test \leftarrow (2>3) ET (2<2);

- Définition de l'opérateur OU

Soit A et B deux expressions booléennes

L'expression A ou B est fausse ssi

A est fausse et B est fausse.

- Exemples

Variables

test: booléen;

...

- test \leftarrow (2>3) ou (2=2) ;
- test \leftarrow (2<3) ou (2=2) ;
- test \leftarrow (2>3) ou (2<2);

- Définition de l'opérateur non
soit A est une expression booléenne.
Si A est vraie, non(A) est fausse.
Si A est fausse, non(A) est vraie.

- Exemples
 - test1 \leftarrow (1<2) ET (2=2) ;
 - test2 \leftarrow non (non(1<2) ou (2=3));
 - test3 \leftarrow non (test2);

- Lois de De Morgan

Si A et B deux expressions booléennes
alors

- $\text{non}(A \text{ ET } B) = \text{non}(A) \text{ OU } \text{non}(B)$
- $\text{non}(A \text{ OU } B) = \text{non}(A) \text{ ET } \text{non}(B)$

- Structures conditionnelles

➤ Forme 1

...

Si (condition) alors

instr1;

instr2;

...

finsi

...

■ Structures conditionnelles

➤ Forme 1

...

Si (condition) alors

instr1;

instr2;

...

finsi;

...

■ Exemple

Chercher la surface d'un disque de rayon saisi au clavier. Il faut s'assurer tout d'abord que le rayon soit positif.

Algorithme surfaceDisque;

Constantes

PI=3.14;

Variables

rayon, surface:réels ;

Début

Lire(rayon);

Si (rayon>0) alors

*Surface \leftarrow rayon*rayon*PI;*

Ecrire("surface=",surface);

Finsi

Fin.

■ Structures conditionnelles

➤ Forme 2

...

Si (condition) alors

instr11;

instr12;

...

Sinon

Instr21;

Instr22;

....

Finsi

...

- Structures conditionnelles

- Forme 2

```
...  
Si (condition) alors  
    instr11;  
    instr12;  
...  
Sinon  
    Instr21;  
    Instr22;  
...  
Finsi;  
...
```

- Exemple

Chercher la surface d'un disque de rayon saisi au clavier.

Il faut s'assurer tout d'abord que le rayon soit positif. Si le rayon saisi est négatif, afficher le message "rayon non valide".

■ Solution

Algorithme surfaceDisque;

Constantes

$PI=3.14$;

Variables

rayon, surface:réels ;

Début

Lire(rayon);

Si (rayon>0) alors

*surface \leftarrow rayon*rayon*PI;*

Ecrire("surface=",surface);

Sinon

Ecrire(" rayon non valide");

Finsi

Fin.

■ Exercices

1/Chercher le minimum de **deux entiers saisis au clavier**.

2/Chercher le minimum de trois entiers saisis au clavier.

3/Donner la solution de $ax+b=0$. a et b sont deux réels à saisir.

▪ Ex1 Solution1

Algorithme minimum1;

Variables

a, b, min :entiers;

Début

Lire(a,b);

Si (a<b) alors

min ← a;

Sinon

min ← b;

Finsi;

Ecrire ("le min est:",min);

Fin.

M.Machkour

▪ Ex1 Solution2

Algorithme minimum2;

Variables

a, b, min :entiers;

Début

Lire(a,b);

min ← a;

Si(min > b) alors

min ← b;

Finsi;

Ecrire ("le min est:",min);

Fin.

Info3-SMP3/2014-1015

93

▪ Ex1 Solution3

Algorithme minimum3;

Variables

a, b, ~~min~~ :entiers;

Début

Lire(a,b);

Si(a<b) alors

Ecrire ("le min est:",a);

Sinon

Ecrire ("le min est:",b);

Finsi

Fin.

M.Machkour

Info3-SMP3/2014-1015

94

■ Ex2 Solution

Algorithme minimum;

Variables

a, b, c, min :entiers;

Début

Lire(a, b, c);

min \leftarrow a;

Si (min > b) alors

min \leftarrow b;

Finsi

Si (min > c) alors

min \leftarrow c;

Finsi

Ecrire ("le min est:",min);

Fin.

M.Machkour

Info3-SMP3/2014-1015

95

■ Ex3 solution

Algorithme equation;

Variables a,b,x :réels;

Début

Lire(a,b);

Si (a \neq 0) alors

X \leftarrow -b/a;

Ecrire("la sol est :", x);

Sinon

Ecrire ("il y a 0 ou plusieurs solutions");

Finsi ;

Fin.

■ Ex 3 Solution complète

Algorithme equation1;

Variables a, b, x :réels;

Début

Lire(a,b);

Si (a≠0) alors

X=-b/a;

Ecrire("la sol est :", x);

Sinon

si (b=0) alors

Ecrire ("il y a plusieurs solutions");

sinon

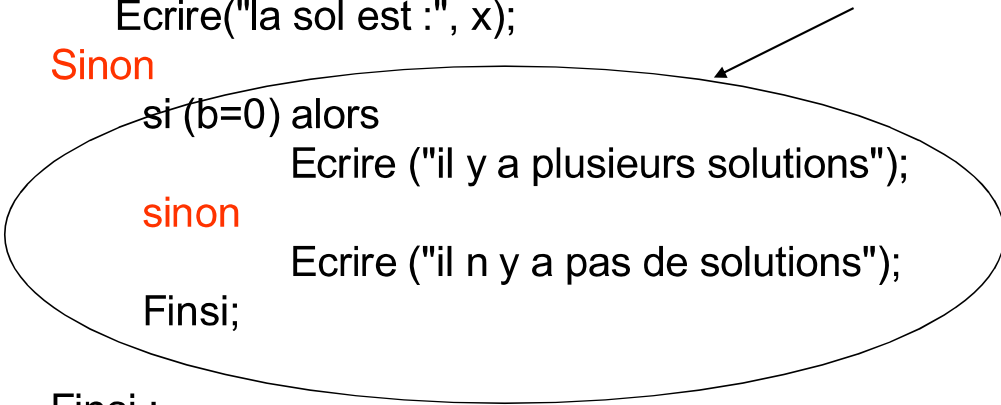
Ecrire ("il n y a pas de solutions");

Finsi;

Finsi ;

Fin.

Structure conditionnelle à l'intérieur
d'une autre → structures
conditionnelles imbriquées



■ Exercices supplémentaires

1/Ecrire un algorithme qui lit la moyenne d'un étudiant et affiche s'il est admis ou non.

2/Ecrire un algorithme qui lit la moyenne d'un étudiant et affiche la mention associée.

3/Ecrire un algorithme qui lit trois notes d'un étudiant, calcule sa moyenne et l'affiche avec la mention associée.

4/Ecrire un algorithme qui lit un entier et affiche s'il est pair ou impair.

5/Ecrire un algorithme qui lit trois nombres et affiche s'ils sont ordonnés ou non.

Ex 2 supp

Algorithme mention;

Variables

moy : réels;

mention : chaîne de caractères;

Début

Lire(moy);

si (moy<12) alors

mention="passable";

sinon

Si (moy<14) alors

mention="A.B";

Sinon

Si (moy<16) alors

mention="B";

sinon

mention="T.B";

Finsi;

Finsi;

Finsi;

Ecrire ("la mention est :", mention);

M. Machkour

Info3-SMP3/2014-1015

99

■ Structures conditionnelles : choix multiple

➤ Forme 3

Selon (expression)

Cas val1 : liste_instructions1;

Cas val2 : liste_instructions2;

...

sinon : Liste_instructions;

Finsel

■ Structures conditionnelles : choix multiple

➤ Forme 3

Selon (expression)

Cas val1 : liste_instructions1;

Cas val2 : liste_instructions2;

...

sinon : Liste_instructions;

Finselon;

■ Fonctionnement de selon

- Si expression=val1 alors on exécute liste_instructions1. Sinon, on passe comparer expression avec vali. Si expression=vali alors seront exécutées liste_instructionsi.
- Si expression est différent de toutes les valeurs vali, on exécute les instructions de sinon si elle est présente. Sinon on passe à l'instruction suivante de l'algorithme.

■ Exemple

Ecrire un algorithme qui lit un opérateur op (+,-,/,*) et deux entiers a et b puis affiche le nom et le résultat de l'opération a op b.

Entrées op, a, b;

Sorties : la valeur de "a op b" et le nom de l'opération.

Algorithme operateur;

Variables

op caractère;

a,b, r : entiers;

Début

Lire(op);

Lire(a,b);

Selon (op)

Cas '+' : $r \leftarrow a+b$; écrire ("la somme de a et b est:", r);

Cas '-' : $r \leftarrow a-b$; écrire ("la différence entre a et b est:", r);

Cas '' : $r \leftarrow a*b$; écrire ("la multiplication de a par b est:", r);*

Cas '/' : $r \leftarrow a/b$; écrire ("la division de a par b est:", r);

Sinon : écrire (op , " : opérateur non valide");

Finselon;

Fin.

■ Les structures itératives

Trois façons pour exprimer les itérations

- Tantque (condition) faire
- Faire ...tantque(condition)
- Pour

■ La structure tantque(condition)

Tantque (condition) faire

Instr1;

Instr2;

...

Fin tantque

■ La structure tantque(condition)

Tantque (condition) faire

Instr1;

Instr2;

...

Fin tantque;

■ La structure tantque(condition)

Fonctionnement

- Si la condition est vraie, on exécute les instrs et on passe vérifie la condition de nouveau. Si elle est toujours vraie, on exécute les instrs et ainsi de suite. Ce processus se répète jusqu'à ce que la condition soit vraie.
- Si la condition est fausse à l'entrée de la boucle, le bloc ne sera jamais exécuté.

■ Remarque

En général, les instrs doivent contenir une **instruction**

qui assure

la **sortie de la boucle** :

rend la condition fausse après un certain nombre fini d'itérations.

■ Exercices

- 1) Écrire un algorithme qui calcule la somme $1+2+3+\dots+n$. n à lire au clavier.
- 2) Écrire un algorithme qui affiche les diviseurs d'un nombre lu au clavier. Utiliser l'opérateur %.

■ Exercice 1

Algorithme Somme;

Var s,n, i : entier;

Début

$s \leftarrow 0;$

$i \leftarrow 1;$

lire(n);

Tantque (i<=n) faire

$s \leftarrow s+i;$

$i \leftarrow i+1;$

fin tantque;

écrire(s);

fin.

■ Exercice 2

Algorithme Diviseurs;

Variables

n, i:entiers;

Début

Lire(n);

$i \leftarrow 1;$

Tantque i<=n faire

Si (n%i)=0 alors

Ecrire(i);

Finsi

$i \leftarrow i+1;$

Fin tantque

Fin .

■ Exercice

Écrire un algorithme qui calcule et affiche la somme des diviseurs d'un nombre lu au clavier. Utiliser l'opérateur %.

■ Réponse

Algorithme SomDiviseurs;

Variables

n, i, som :entiers;

Début

Lire(n);

i ← 1;

som ← 0;

Tantque (i ≤ n) faire

Si (n % i) = 0 alors

som ← som + i;

Finsi

i ← i + 1;

Fin tantque

Ecrire ("la somme des diviseurs est :", som);

Fin .

- La structure faire ... tantque(condition)

Faire

Instr1;

Instr2;

...

Tantque (condition)

- La structure faire ... tantque(condition)

Faire

Instr1;

Instr2;

...

Tantque (condition);

■ La structure faire ... tantque(condition)

Fonctionnement

- On exécute les instructions délimitées par *Faire* et *Tantque* , puis on vérifie la condition.
- Si la condition est vraie, on réexécute les instrs et on passe vérifie la condition de nouveau. Si elle est encore vraie, on exécute les instrs et ainsi de suite. Ce processus se répète tant que la condition est vraie.
- Une fois la condition est fausse on quitte la structure.

■ La structure faire ... tantque(condition)

Remarques

- En général, les instrs doivent contenir une instruction qui assure la sortie de la boucle: rendre la condition fausse après un certain nombre fini d'itérations ou autre .
- Les instructions contrôlées par cette structure sont exécutées au moins une fois.

■ Exemple

Algorithme Somme;

Var s,n, i : entier;

Début

$s \leftarrow 0;$

$i \leftarrow 1;$

lire(n);

Faire

$s \leftarrow s+i;$

$i \leftarrow i+1;$

Tantque (i<=n)

ecrire(s);

fin.

■ La structure pour

Pour var_compteur \leftarrow début **à** fin **faire**

Intsr1;

Instr2;

..

← Ce bloc sera exécuté fin-début+1 fois

Finpour

■ La structure pour

Pour var_compteur \leftarrow début **à** fin **faire**

Intsr1;
Instr2;
..
Ce bloc sera exécuté fin-début+1 fois

Finpour;

■ La structure pour

Fonctionnement de la structure

- i) On affecte à var_compteur la valeur de début, puis on exécute les instructions.
- ii) Ensuite, on incrémente var_compteur et on vérifie si sa valeur est \leq de celle de fin.
- iii) Si var_compteur \leq fin, on exécute les instructions et on passe à ii)

On quitte cette boucle lorsque var_compteur $>$ fin.

■ La structure pour

Remarque

var_compteur, début et fin doivent avoir le même type ou des types compatibles.

Exemple

Calculer la somme $s=1+2+3..+n$. n à lire au clavier

■ Exemple

Algorithme Somme;

Variable s,n,i : entier;

Début

Lire(n);

s ← 0;

pour i ← 1 à n faire

s ← s+i;

finpour

Ecrire(s);

fin.

Cas de n=3

- $i=1, i \leq n = 3 ? \Rightarrow s \leftarrow s+i \Rightarrow s \leftarrow 0+1 \Rightarrow s=1$
- i devient i+1, donc $1+1=2$
- $i=2 \leq n=3 ? s \leftarrow s+i \Rightarrow s \leftarrow 1+2 \Rightarrow s=3$
- i devient i+1, donc $2+1=3$
- $i=3 \leq n=3 ? \Rightarrow s \leftarrow s+i \Rightarrow s \leftarrow 3+3 \Rightarrow s=6$
- i devient i+1, donc $3+1=4$
- $i=4 > n=3$, on quitte la boucle.

■ Exercice

Chercher la somme des diviseurs d'un entier saisi au clavier.

Réponse

```
Algorithme SomDiv;  
Variable s,n,i : entier;  
Début  
    s ← 0;  
    Lire(n);  
    pour i ← 1 à n faire  
        Si (n%i)=0 alors  
            s ← s+i;  
        Finsi  
    finpour  
    Ecrire(s);  
fin.
```

■ Exercice

Ecrire un algorithme qui calcule le factoriel d'un entier saisi au clavier.

```
Algorithme factoriel;  
Variable fact,n,i : entier;  
Début  
    Lire(n);  
    fact ← 1;  
    pour i ← 1 à n faire  
        fact ← fact*i;  
    finpour  
    Ecrire(fact);  
Fin.
```

■ Exercice

Écrire un algorithme qui lit un entier et affiche s'il est pair ou non.

Algorithme parité;

Variable n:entier;

Début

Lire(n);

Si $(n \% 2) = 0$ alors

Ecrire(pair);

Sinon

Ecrire (impair);

Finsi

Fin.

■ Organigramme

Définition

- Organigramme = Représentation graphique d'un algorithme.
- Chaque instruction et chaque structure contrôle possède une représentation graphique.

■ Organigramme

Affectation

Lecture

Ecriture

Début

Fin

Séquence

Reste de l'algorithme ○ (avant ou après l'instruction en cours)

surface \leftarrow PI*rayon*rayon

Lire (rayon)

Ecrire (surface)

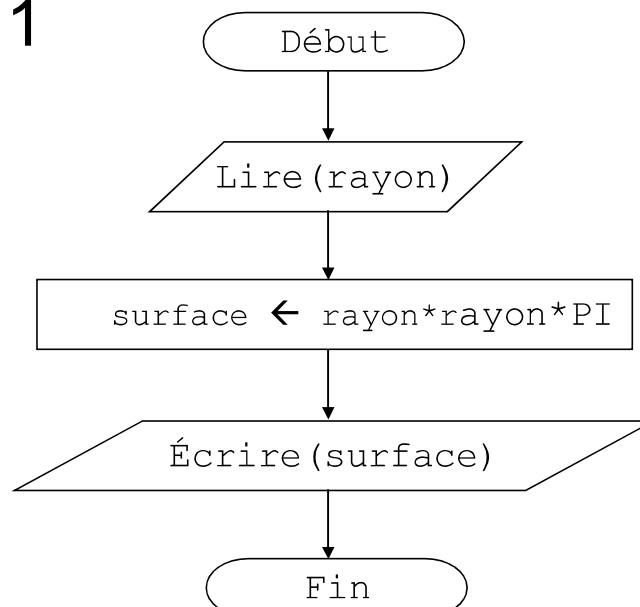
Début

Fin



■ Organigramme

Exemple 1

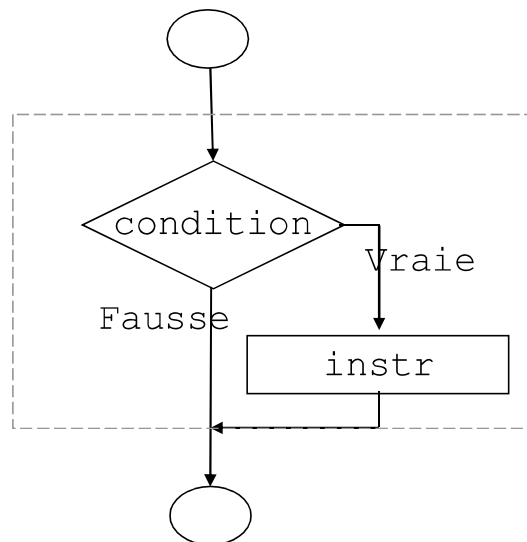


■ Organigramme

Si (condition) alors

instr;

Finsi



■ Organigramme

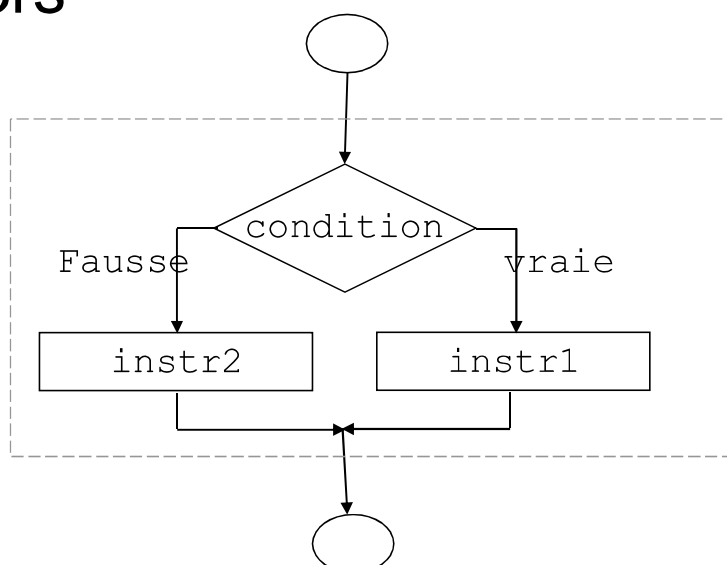
Si (condition) alors

instr1;

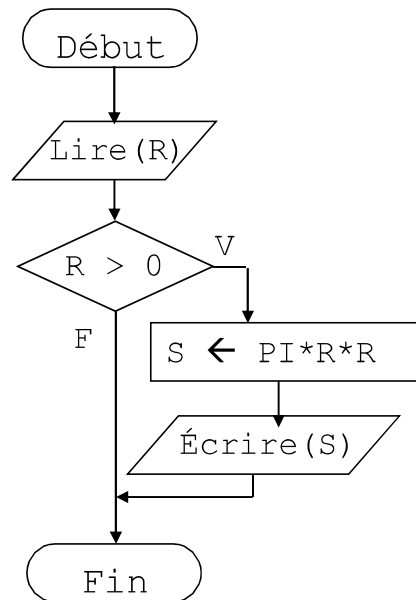
Sinon

instr2;

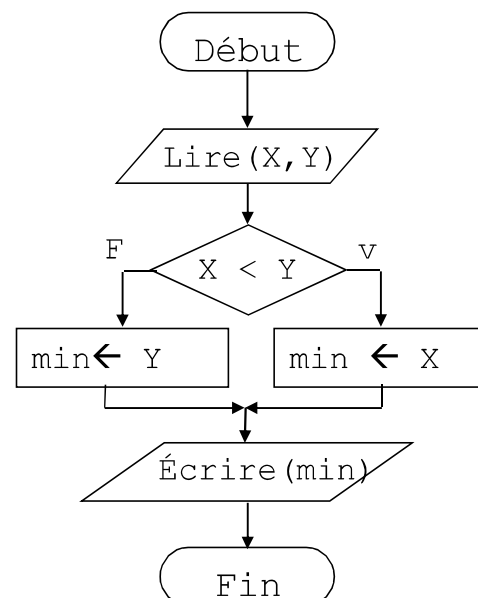
Finsi



■ Organigramme Exemple 2



■ Organigramme Exemple 3 Recherche de minimum

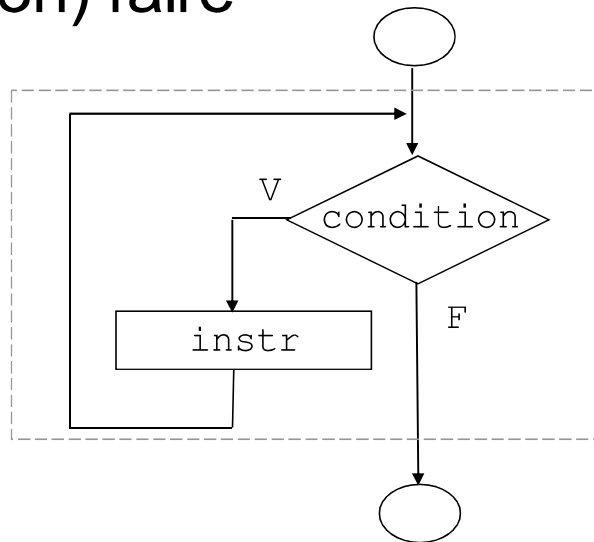


■ Organigramme

Tantque (condition) faire

instr;

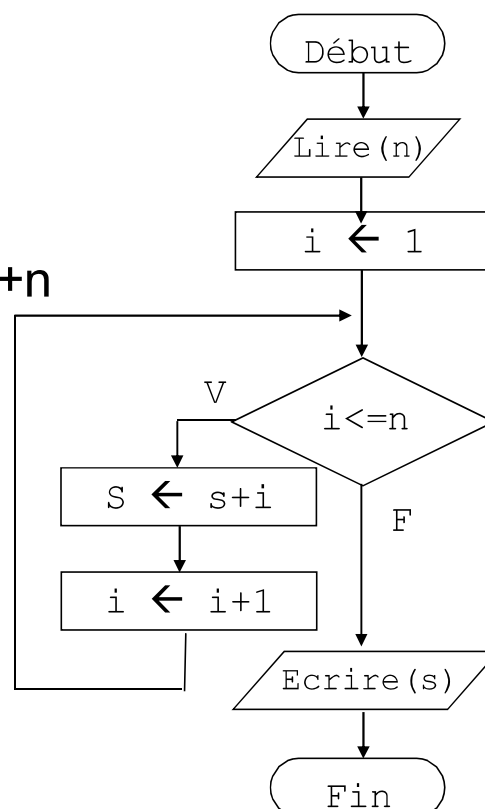
Fin tantque



■ Organigramme

Exemple 4

Calcul de $S=1+2+3+\dots+n$

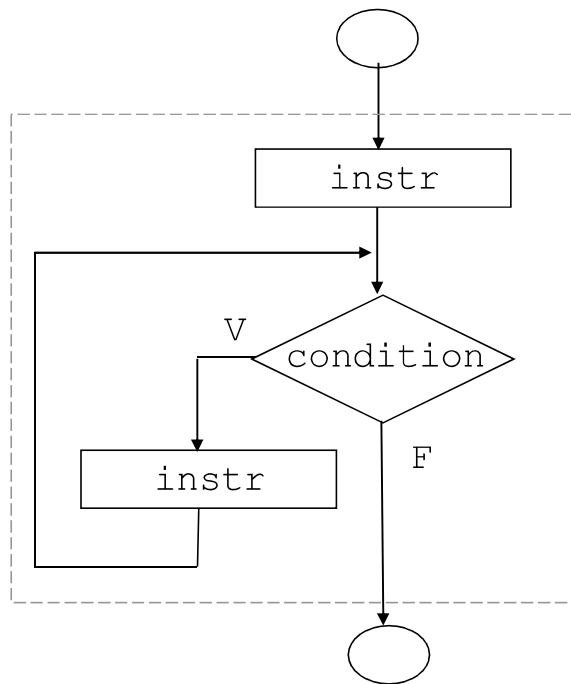


■ Organigramme

Faire

instr;

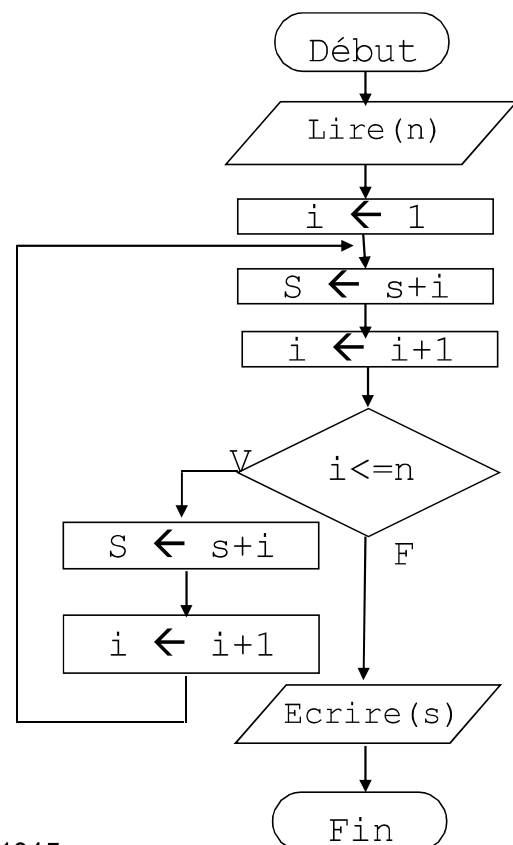
Tantque (condition)



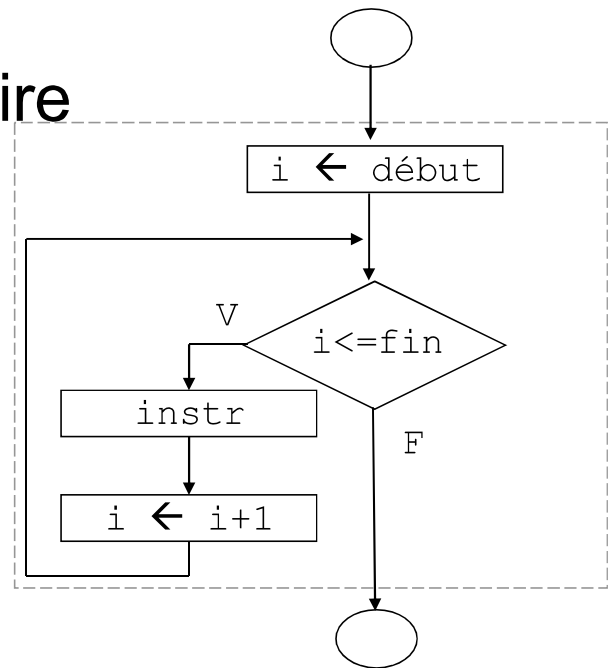
■ Organigramme

Exemple 5

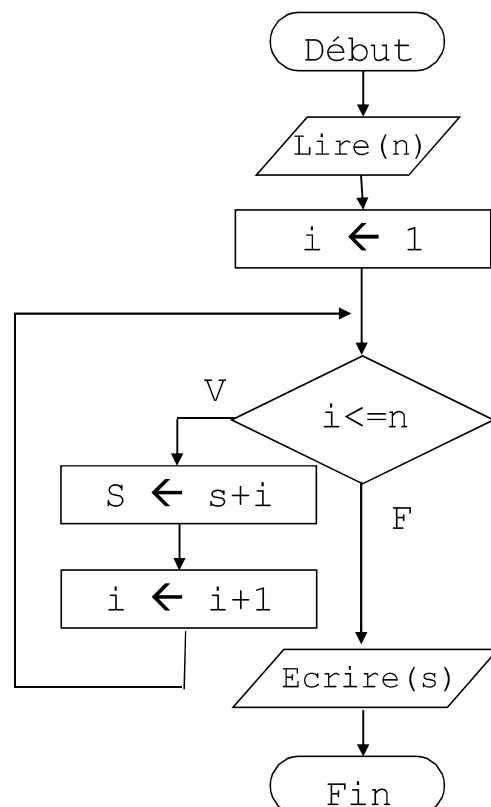
Calcul de $S=1+2+3...+n$



- Organigramme
pour $i \leftarrow \text{début}$ à fin faire
instr;
Finpour



- Organigramme
Exemple 6
Calcul de $S=1+2+3\dots+n$



Fin.