

### Deuxième partie



#### I) Notions de base

##### A) Les variables et les constantes

Dans la syntaxe des pseudo-codes et organigrammes *LARP*, une *constante* est **numérique** ou **alphanumérique**. Par exemple, **12.3** est une constante numérique représentant un nombre fractionnel, et "**allo**" est une constante alphanumérique représentant une chaîne de caractères. *LARP* supporte plusieurs types de constantes.

Une *variable* dans un algorithme *LARP* (comme dans la plupart des langages de programmation tels *C++* et *Java*) est un emplacement dans la mémoire de l'ordinateur où sont stockées des données.

Comme dans tous les langages de programmation, des règles précises régissent la sélection des noms de variables :

- Un nom de variable doit débuter par une lettre (**A** à **Z**, **a** à **z**) ou le caractère de soulignement (**\_**).
- Un nom de variable peut être constitué de lettres minuscules (**a** à **z**), de lettres majuscules (**A** à **Z**), de chiffres (**0** à **9**) et du caractère de soulignement (**\_**).
- Un nom de variable ne doit pas correspondre à un mot réservé de *LARP*, tels que **ÉCRIRE**, **FIN**, **SI** et **PI**, et ce sans égard aux accents (par exemple **ÉCRIRE** est aussi un mot réservé).

##### B) Les opérateurs relationnels

Opérateurs	Descriptions
<b>&lt;</b>	Plus petit : $a < b$
<b>&lt;=</b>	Plus petit ou égal : $a \leq b$
<b>&gt;</b>	Plus grand : $a > b$
<b>&gt;=</b>	Plus grand ou égal : $a \geq b$
<b>=</b>	Égalité : $a = b$
<b>!=</b>	Inégalité : $a \neq b$ . Le symbole équivalent <b>&lt;&gt;</b> est aussi supporté par <i>LARP</i> .

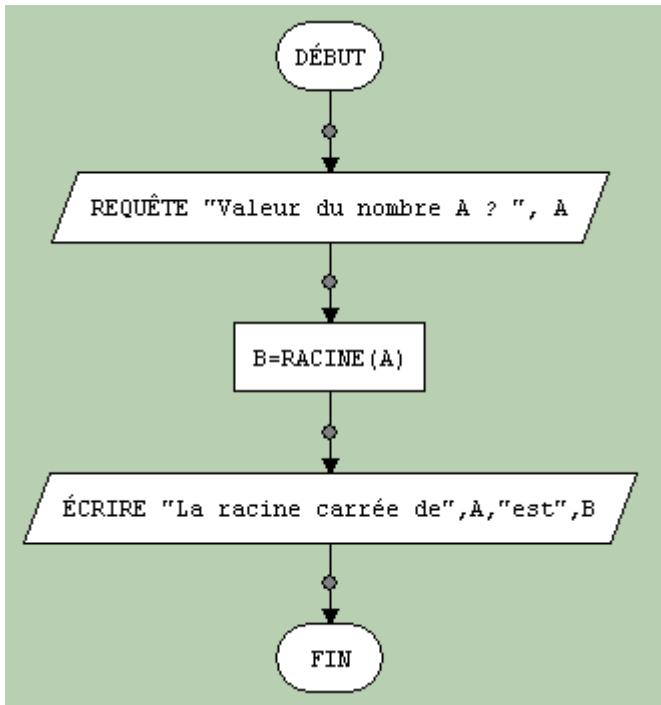
Remarque : Dans la plupart des logiciels, **different** s'écrit **<>**

## II) Les structures conditionnelles

### A) Les limites d'une structure linéaire

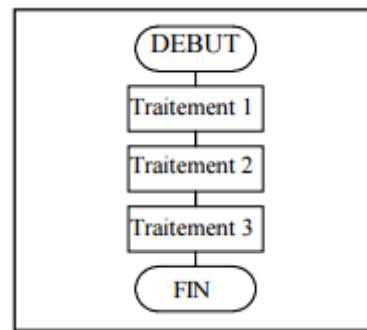
#### Application 1

Reproduire l'organigramme suivant permettant de calculer la racine carrée d'un nombre.



#### RAPPEL

La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre de leur énoncé.

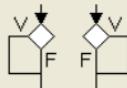


- Faire un essai avec le nombre 120
- Faire un essai avec le nombre -120
- Enregistrer le travail (nom du fichier : racine1)
- Conclure



## B) La structure conditionnelle SI

La plupart des problèmes à résoudre par programmation comporte l'éventualité d'effectuer un choix dans l'algorithme. Une *structure conditionnelle* est une instruction permettant de spécifier des séquences d'instructions alternatives dans un algorithme.

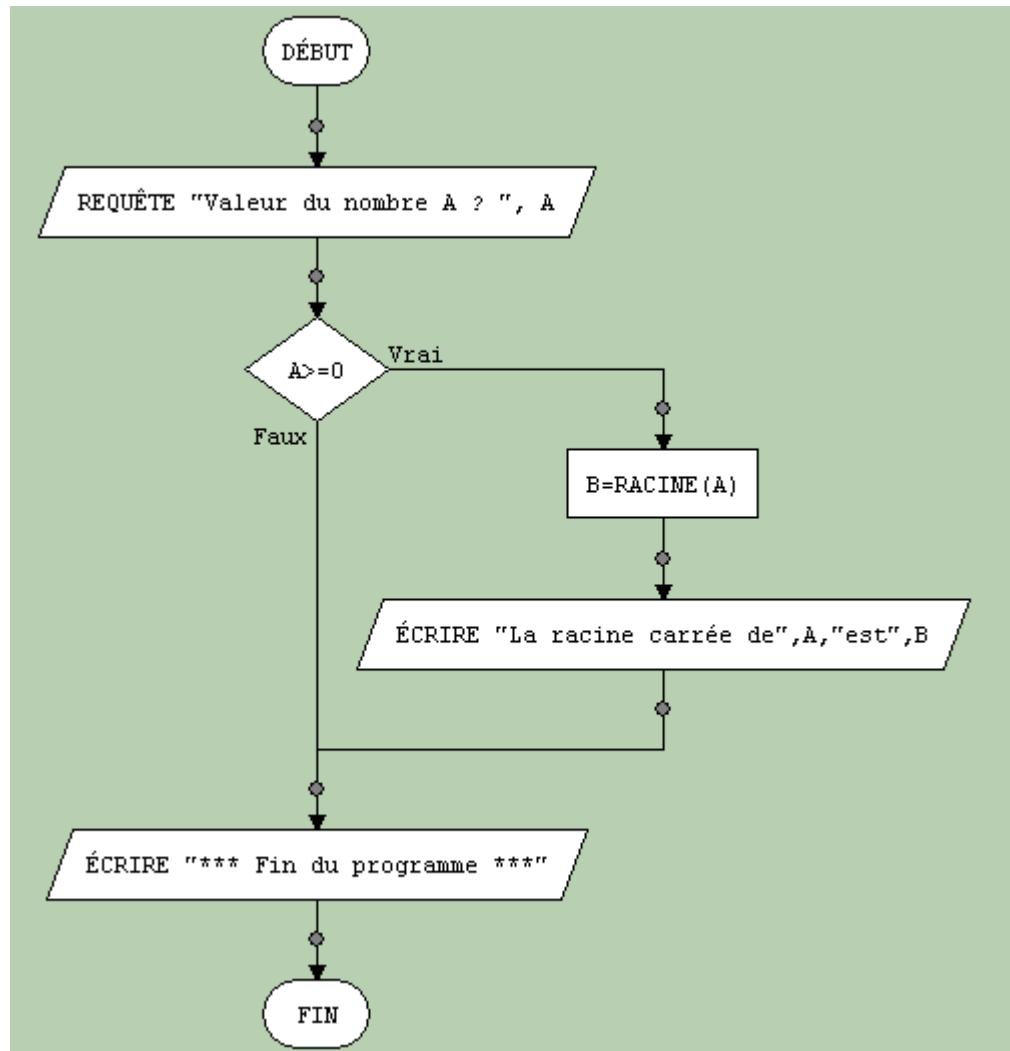


**Structure conditionnelle SI** : une **séquence d'instructions** exécutée uniquement en fonction du résultat de l'évaluation d'une **condition**.

### Application 2

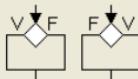
Reproduire l'organigramme suivant permettant de calculer la racine carrée d'un nombre.  
Si le nombre proposé est négatif, aller à la fin du programme.

Enregistrer le travail (nom du fichier : racine2)



Conclure :

### C) La structure conditionnelle SI-ALORS-SINON (Si-SINON)



**Structure conditionnelle *SI-S/NON*** : deux séquences d'instructions dont une et une seule est exécutée en fonction du résultat de l'évaluation d'une **condition**.

Cette structure est plus souvent appelée **SI-ALORS-SINON**

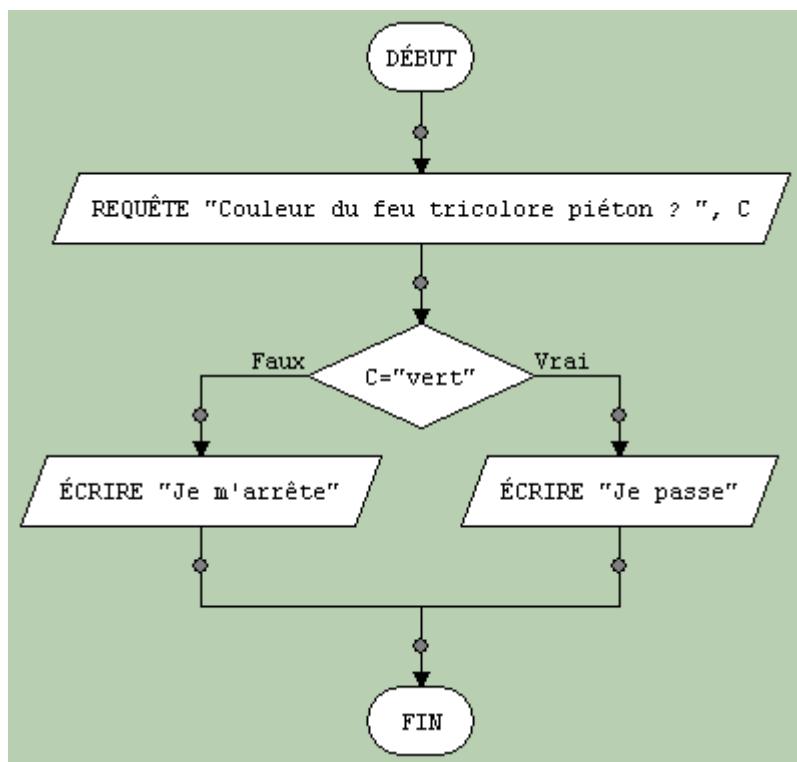
Dans ce cas de figure on a le choix entre deux séquences d'instructions mais une seule seulement sera exécutée en fonction d'une **condition**.

#### Application 3

Reproduire l'organigramme suivant et tester celui-ci :

Enregistrer le fichier (nom du fichier : feu tricolore)

Les réponses seront écrites en minuscules



**SI** [le feu tricolore est vert]

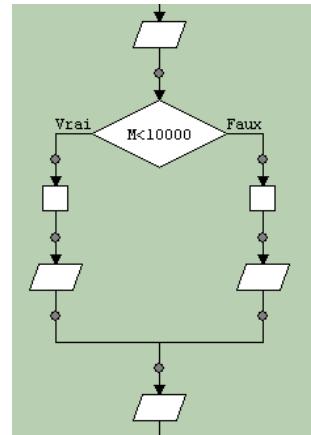
**ALORS** [je passe],

**SINON** [je m'arrête.]

#### Application 4

On vous fournit le pseudo-code suivant réalisé avec LARP ; pseudo-code permettant de calculer une remise en fonction du montant de la commande :

```
DÉBUT
REQUÊTE "Montant commande ? ", M
SI M<10000 ALORS
    R=M*3/100
    ÉCRIRE "Taux de remise : 3 %"
SINON
    R=M*5/100
    ÉCRIRE "Taux de remise : 5 %"
FINSI
ÉCRIRE "Remise", R,"euros"
FIN
```

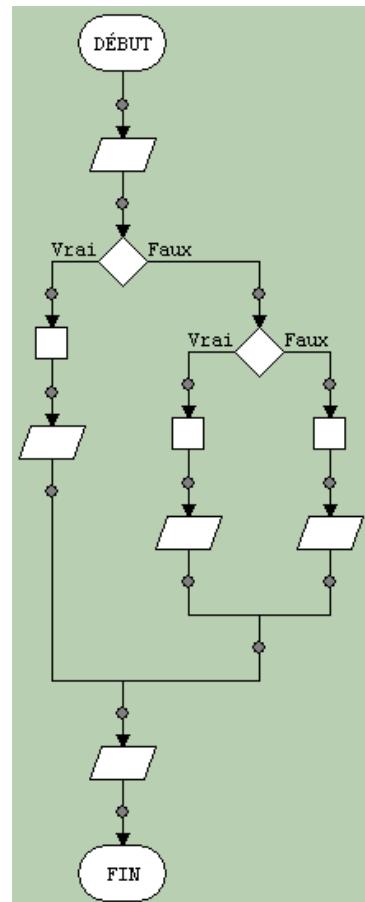


Créer l'organigramme correspondant sur LARP et tester celui-ci ( nom du fichier : remise1)

#### Application 5

On vous fournit le pseudo-code suivant réalisé avec LARP ; pseudo-code permettant de calculer une remise en fonction du montant de la commande :

```
DÉBUT
REQUÊTE "Montant commande ? ", M
SI M<10000 ALORS
    R=M*3/100
    ÉCRIRE "Taux de remise : 3 %"
SINON
    SI M<20000 ALORS
        R=M*5/100
        ÉCRIRE "Taux de remise : 5 %"
    SINON
        R=M*7/100
        ÉCRIRE "Taux de remise : 7 %"
    FINSI
    FINSI
ÉCRIRE "Montant de la remise :", R,"euros"
FIN
```



Créer l'organigramme correspondant sur LARP et tester celui-ci

Enregistrer le travail (Nom du fichier : remise2)

```
Montant commande ? 8000
Taux de remise : 3 %
Montant de la remise : 240 euros
```

```
Montant commande ? 14000
Taux de remise : 5 %
Montant de la remise : 700 euros
```

Tester  
successivement  
ces 3  
hypothèses

```

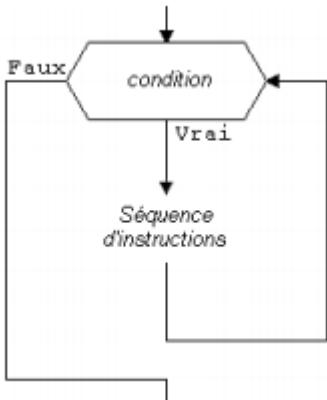
Montant commande : 24000
Taux de remise : 7 %
Montant de la remise : 1680 euros

```

### III) Les boucles

Dans un algorithme, utiliser une boucle permet de recommencer plusieurs fois un bloc d'instructions.

#### A) La structure TANTQUE... FAIRE (ou TANT QUE... FAIRE)



#### Structure répétitive TANTQUE

**TANTQUE condition FAIRE**  
*Séquence d'instructions*  
**FINTANTQUE**

La structure **TANT QUE...** permet de répéter une série d'instructions **tant qu'une certaine condition est vérifiée**.

#### Application 6

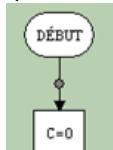
On lance une balle d'une hauteur initiale de 300 cm.

On suppose qu'à chaque rebond, la balle perd 20 % de sa hauteur (la hauteur est donc multipliée par 0.8 à chaque rebond).

On cherche à savoir le nombre de rebonds nécessaires pour que la hauteur de la balle soit inférieure ou égale à 10 cm.

**H** est la variable contenant la hauteur ;  
**C** est la variable contenant le compteur.

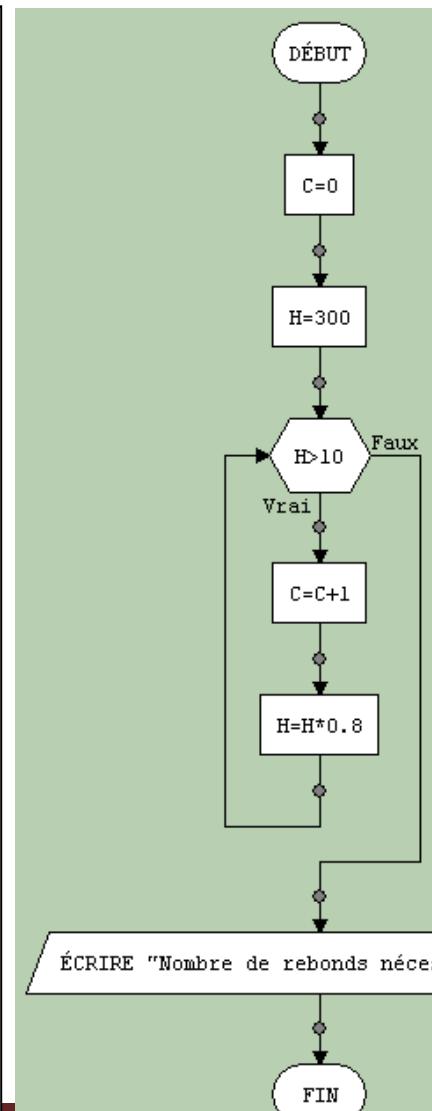
Remarque : Lorsque l'on fait un calcul du type  $C=C+1$ , il faut remettre la variable à sa valeur initiale au début du programme pour que celui-ci fonctionne ( $C=0$  dans ce cas).



Reproduire le logigramme ci-contre avec LARP



Enregistrer le travail  
(Nom du fichier : rebonds)



**C=C+1** veut dire que la **nouvelle valeur** de **C** est égale à l'**ancienne valeur** de **C** à laquelle on rajoute **1**.

Dans ce cas, la variable **C** est un **compteur**.

## Application 7

On place un capital de 500 € sur un compte rémunéré à 3 % par an.

L'algorithme permet de calculer le nombre d'années au bout desquelles le capital sera doublé.

Il y a 2 variables dans ce cas :

- **MONTANT** : montant du capital ;
- **COMPTEUR** : permet de compter le nombre des années.

On vous fournit le pseudo-code suivant réalisé avec LARP ; pseudo-code permettant de calculer une remise en fonction du montant de la commande :

DÉBUT

```
COMPTEUR=0  
MONTANT=500  
TANTQUE MONTANT<=1000 FAIRE  
    MONTANT=MONTANT*1.03  
    COMPTEUR=COMPTEUR+1  
FINTANTQUE  
ÉCRIRE "Le montant est de",ARRONDIR(MONTANT),"euros"  
ÉCRIRE "après",COMPTEUR,"années de placement"  
FIN
```

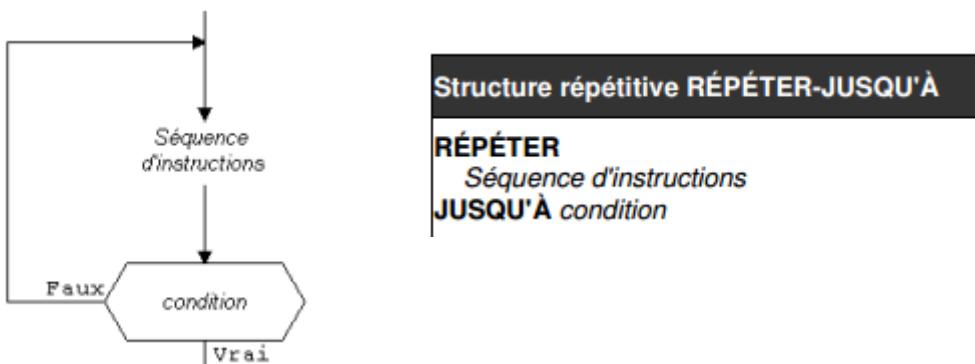


Créer le logigramme correspondant sur LARP et tester celui-ci (nom du fichier : intérêts)

### B) La structure REPETER-JUSQU'A

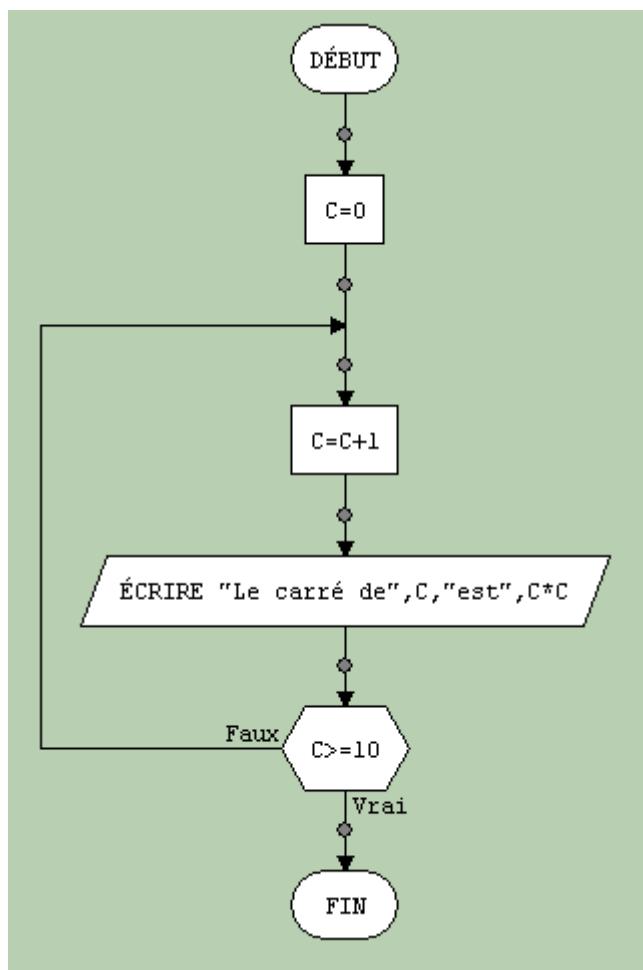
La structure **RÉPÉTER-JUSQU'À** est semblable à la structure **TANTQUE**. Comme cette dernière, elle inclut l'exécution d'une *séquence d'instructions* à répétition et en fonction de la valeur d'une *condition*. Cependant, les structures RÉPÉTER-JUSQU'A et TANTQUE diffèrent sur deux points :

1. La structure TANTQUE exécute la séquence d'instructions tant et aussi longtemps que la condition est satisfaite, alors que la structure RÉPÉTER-JUSQU'À exécute la séquence d'instructions tant et aussi longtemps que la condition *n'est pas* satisfaite. En d'autres mots, la structure RÉPÉTER-JUSQU'À itère *jusqu'à* ce que la condition devienne vraie.
2. La structure TANTQUE vérifie la condition *avant* chaque *itération*, alors que la structure RÉPÉTER-JUSQU'À vérifie la condition *après* chaque *itération*.



## Application 8

On vous fournit le logigramme suivant réalisé avec LARP :



Compléter :

**Au début, la valeur de la variable C est égale à  
Celle-ci augmente de  
On calcule le  
On répète l'opération jusqu'à ce que la valeur de C**

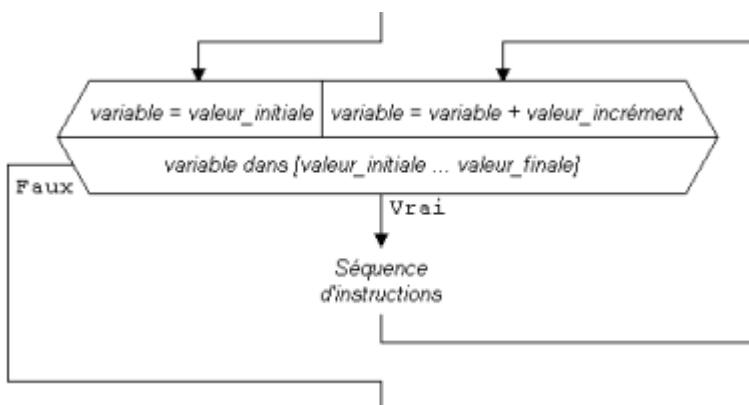
Reproduire l'organigramme suivant et tester celui-ci :

Enregistrer le fichier (nom du fichier : carré)

### C) La structure POUR (POUR... DE... A...)

Il est fréquent que le **nombre de répétitions** soit **connu** à l'avance, et que l'on ait besoin d'utiliser le numéro de l'itération afin d'effectuer des calculs ou des tests. Le mécanisme permettant cela est la **boucle POUR**.

Cette boucle permet de **parcourir un intervalle** en répétant un traitement pour chacune des valeurs de cet intervalle



#### Structure répétitive POUR

```
POUR variable = valeur_initiale JUSQU'À valeur_finale INCRÉMENT valeur_incrément FAIRE  
    Séquence d'instructions  
FINPOUR
```

#### Application 9

Le chiffre d'affaires de l'entreprise ALPHA est estimé à 200 000 € en 2017.  
Celui-ci augmentera en moyenne de 15 500 € tous les ans.  
On désire connaître le montant du chiffre d'affaires de 2017 à 2026.

On va créer un algorithme avec LARP.

Les 2 variables seront :

A : Année  
C : Montant du chiffre d'affaires.

Pour la variable A,

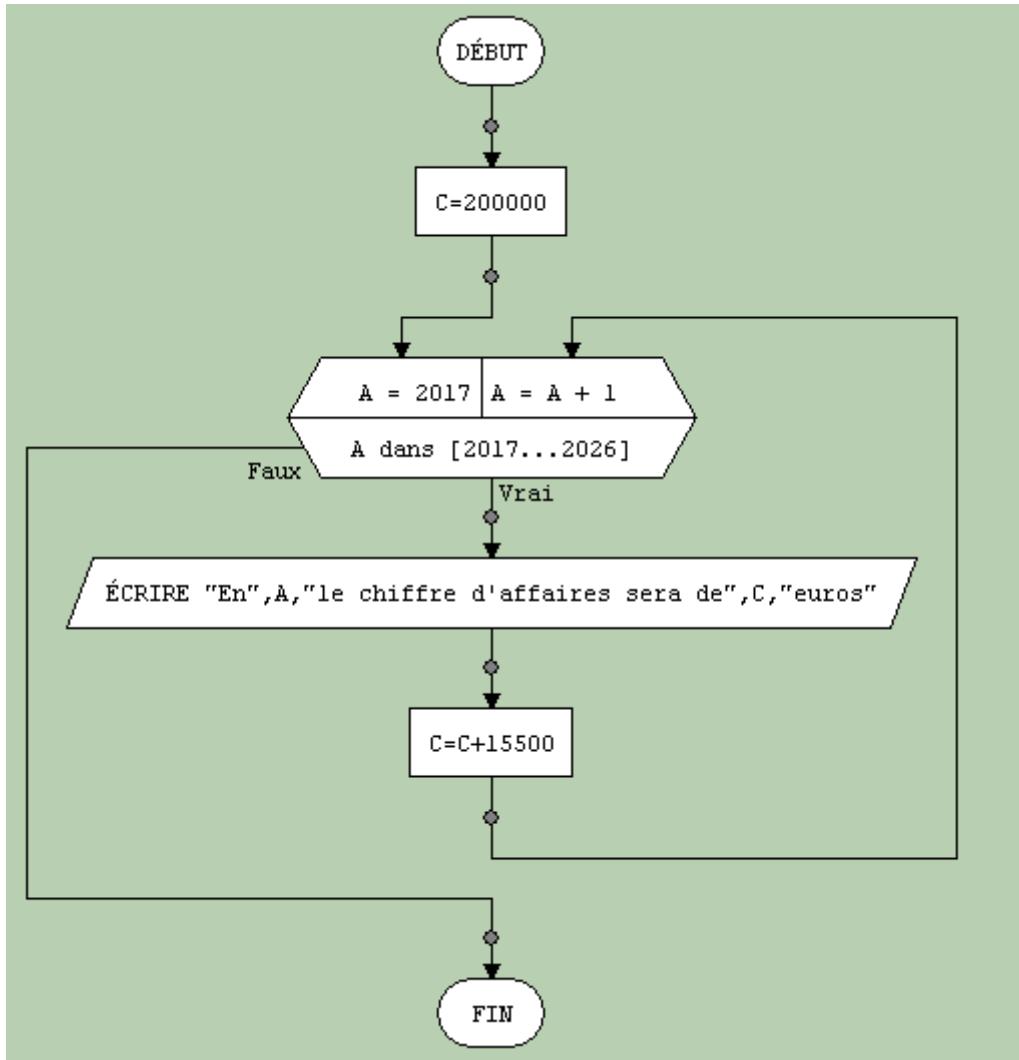
- L'incrément est de 1 : On peut dire que A=
- La valeur initiale est de
- La valeur finale est de

Pour la variable C, chaque année :

- C=

Reproduire l'organigramme suivant et tester celui-ci :

Enregistrer le fichier (nom du fichier : ventes) 



### Application 10

A partir d'un algorithme créé avec LARP, on désire obtenir le résultat suivant :

```

La racine carrée de 0 est 0
La racine carrée de 4 est 2
La racine carrée de 8 est 2.82842712474619
La racine carrée de 12 est 3.46410161513775
La racine carrée de 16 est 4
La racine carrée de 20 est 4.47213595499958
La racine carrée de 24 est 4.89897948556636
La racine carrée de 28 est 5.29150262212918
La racine carrée de 32 est 5.65685424949238
La racine carrée de 36 est 6
  
```

N désignera la variable ; sa racine carrée de N est R.

Vous devez retrouver la valeur minimale de N, la valeur maximale de N et l'incrément.

Enregistrer le travail (nom du fichier : racine3)



### Application 11

On désire créer un jeu avec LARP.

LARP tire au sort un nombre aléatoire compris entre 1 et 1000 mais n'affiche pas celui-ci.

```
J'ai tiré au sort un nombre entre 1 et 1000. A vous de le retrouver  
Proposez un nombre :
```

Objectif : l'utilisateur du programme doit découvrir le nombre en un minimum d'essais

- Si le nombre entré par l'utilisateur n'est pas le bon, le message « Trop grand » ou « Trop petit » devra apparaître. Dans ce cas, l'utilisateur devra proposer un autre nombre.
- Si le nombre entré par l'utilisateur est le bon, le message : "Bravo. Vous avez trouvé en",C,"essais"

Exemple :

- A est le nombre aléatoire tiré au sort par LARP (*Dans l'exemple ci-dessous, A=273*)
- B est le nombre donné par l'utilisateur (Tant que A n'est pas trouvé, LARP demande la valeur de B)
- C est le nombre d'essais qu'il a fallu à l'utilisateur pour retrouver le nombre A.

```
J'ai tiré au sort un nombre entre 1 et 1000. A vous de le retrouver  
Proposez un nombre : 500  
Trop grand  
Proposez un nombre : 250  
Trop petit  
Proposez un nombre : 325  
Trop grand  
Proposez un nombre : 290  
Trop grand  
Proposez un nombre : 270  
Trop petit  
Proposez un nombre : 280  
Trop grand  
Proposez un nombre : 275  
Trop grand  
Proposez un nombre : 273  
Bravo. Vous avez trouvé en 8 essais  
  
Appuyez sur une touche pour fermer la console..._
```

Enregistrer le fichier (nom du fichier : jeu nombre)





## Application 12

Soit la fonction  $f(x)=3x^2-4x+8$

On désire créer le logigramme qui permet d'afficher les valeurs de x et de y pour les valeurs de x comprises entre -10 et 10.

Dans tous les cas, x est un entier relatif (x=-10 puis x=-9, etc... jusqu'à x=10)

```
x= -10 y= 348
x= -9 y= 287
x= -8 y= 232
x= -7 y= 183
x= -6 y= 140
x= -5 y= 103
x= -4 y= 72
x= -3 y= 47
x= -2 y= 28
x= -1 y= 15
x= 0 y= 8
x= 1 y= 7
x= 2 y= 12
x= 3 y= 23
x= 4 y= 40
x= 5 y= 63
x= 6 y= 92
x= 7 y= 127
x= 8 y= 168
x= 9 y= 215
x= 10 y= 268
```



Enregistrer le fichier (nom du fichier : fonction)

## Application 13

On prend la liste des nombres entiers compris entre 14 898 et 16 023.

Parmi cette liste, on désire n'afficher que les nombres multiples de 57 mais qui ne sont pas multiples de 5.

Construire le logigramme correspondant. La fonction **ARRONDIR** devra être utilisée.

- Remarque 1 : La calculatrice est interdite pour réaliser ce travail.
- Remarque 2 : Le nombre 14934 doit être retrouvé et non saisi.

```
14934 est un multiple de 57 et pas de 5
14991 est un multiple de 57 et pas de 5
15048 est un multiple de 57 et pas de 5
15162 est un multiple de 57 et pas de 5
15219 est un multiple de 57 et pas de 5
15276 est un multiple de 57 et pas de 5
15333 est un multiple de 57 et pas de 5
15447 est un multiple de 57 et pas de 5
15504 est un multiple de 57 et pas de 5
15561 est un multiple de 57 et pas de 5
15618 est un multiple de 57 et pas de 5
15732 est un multiple de 57 et pas de 5
15789 est un multiple de 57 et pas de 5
15846 est un multiple de 57 et pas de 5
15903 est un multiple de 57 et pas de 5
16017 est un multiple de 57 et pas de 5
*****
Nombre de réponses trouvées : 16
*****
```



Enregistrer le fichier (nom du fichier : multiples)