

Algorithmique

Syntaxe

Un algorithme est écrit en utilisant une vingtaine de mots réservés. La syntaxe algorithmique utilise un pseudo-code assez proche de la syntaxe d'un langage de programmation tel que le Pascal, C, PHP...

Forme générale d'un algorithme

- Un algorithme est une liste d'actions réalisées sur des données...



✓ Préciser les différentes parties d'un algorithme

Forme générale d'un algorithme (1)

- Un algorithme est de la forme :

Algorithme Nom_algorithme

Partie Déclarations

Partie Instructions

- Plusieurs instructions peuvent être rassemblées pour former une instruction composée ou bloc qui est délimité par les mots clés *début* et *fin*.

- en C :

- une instruction simple est suivie par le terminateur ';'
- un bloc d'instructions est encadré par une paire d'accolades :
{
 bloc d'instructions
}

Forme générale d'un algorithme (2)

- Exemple de calcul de périmètre d'un cercle :

Algorithme Périmètre_Cercle

/ Rôle : Déterminer le périmètre d'un cercle */*

Constante

PI <- 3.1416 : réel

Variable

rayon, périmètre : réel

Début

/ Saisir le rayon au clavier */*

/ Attention : On ne fait aucun contrôle sur la saisie */*

Ecrire("Rayon : ")

Lire(rayon)

/ Calculer le périmètre */*

*périmètre <- 2*PI*rayon*

/ Afficher le périmètre */*

Ecrire("Le périmètre est : ", périmètre)

Fin

Forme générale d'un algorithme (3)

□ Exemple de calcul de périmètre d'un cercle (en C):

```
/* Rôle : Déterminer le périmètre d'un cercle */
#include <stdio.h>
void main()
{
    const float PI = 3.1416 ;
    double rayon, perimetre;
    /* Saisir le rayon au clavier */
    /* Attention : On ne fait aucun contrôle sur la saisie */
    printf("Rayon = ");
    scanf("%lf", &rayon);
    /* Calculer le périmètre */
    perimetre = 2 * PI * rayon; /* par définition */
    /* Afficher le périmètre */
    printf("Le périmètre est : %4.2f\n", perimetre);
}
```

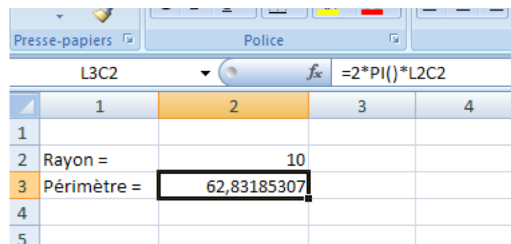
Forme générale d'un algorithme (4)

□ Exemple de calcul de périmètre d'un cercle (en VBA) :

```
Sub perimetre_cercle()
Const PI As Double = 3.1416
Dim rayon As Double
Dim perimetre As Double
'Saisir le rayon dans une fenêtre de saisie
rayon = InputBox("Rayon")
'Calculer le périmètre
perimetre = 2 * PI * rayon
'Afficher le périmètre dans une fenêtre
MsgBox ("Le périmètre est : " & perimetre)
End Sub
```

Forme générale d'un algorithme (5)

- Exemple de calcul de périmètre d'un cercle (avec un tableur) :



	1	2	3	4
1				
2	Rayon =	10		
3	Périmètre =	62,83185307		
4				
5				

Forme générale d'un algorithme (6)

- Une action à effectuer peut correspondre soit à une instruction simple, soit à un bloc d'instructions
- Une action peut être conditionnée. La condition est une expression écrite entre parenthèses et ayant une valeur booléenne : VRAI ou FAUX
- Les actions conditionnées sont mises en œuvre par des structures de contrôle.
Exemple : Tester que le rayon est positif avant de calculer le périmètre.

Les commentaires

- ❑ Les commentaires sont obligatoires pour plus de lisibilité...



✓ Insérer des commentaires

Les Commentaires (1)

- ❑ Les commentaires servent à clarifier un algorithme en donnant des explications. Ces lignes de textes sont généralement précédées (ou encadrées) par des symboles spéciaux qui signaleront au compilateur de les ignorer.

- ❑ Un commentaire débute par /* et se termine par */.

Exemple :

/ Ceci est un commentaire */*

Les Commentaires (2)

- Il est conseillé de faire précéder le corps de l'algorithme par un commentaire en-tête explicitant au moins la fonction de l'algorithme, les données en entrée, les données en sortie.
- On pourra aussi préciser des informations tels que le nom de l'auteur, la version de l'algorithme, la date de création de l'algorithme...

Les variables

- Le résultat d'un algorithme dépend des variables qu'il utilise...



✓ Définir et caractériser une variable

Les Variables (1)

- Les données sont stockés dans des variables
- Les données peuvent être de plusieurs types
- **Déclaration d'une variable :**
 - un nom
 - le type de données qu'elle contient.
- Toutes les variables utilisées dans un algorithme doivent être déclarées.

Les Variables (2)

- **Types de données simples :**
 - entier
Déclaration algorithmique : *n1, n2 : entier*
Déclaration C : *int n1, n2;*
 - réel
Déclaration algorithmique : *x1, x2 : réel*
Déclaration C : *float a, b;*
 - Caractère
Déclaration algorithmique : *c1, c2 : caractère*
Déclaration C : *char c1, c2 ;*
 - booléen
{VRAI, FAUX}
Déclaration algorithmique : *a, b : booléen*
Déclaration C : *n'existe pas - Il faut le définir.*

Le type Chaîne caractérise les chaînes de caractères.

Les Variables (3)

- Le type *int* en C dépend de la taille du mot utilisé par le processeur. Il peut être qualifié par un modificateur pour fixer sa taille et préciser l'ensemble des valeurs :
 - *int* (4 octets) : [-2147483648..2147483647]
(sur une machine 32 bits)
 - *short int* (2 octets) : [-32768..32767]
 - *unsigned short int* (2 octets) : [0..65535]
 - *unsigned int* (4 octets) : [0..4294967295]
 - *long int* (4 octets) : [-2147483648..2147483647]
 - *unsigned long int* (4 octets) : [0..4294967295]
- Il existe deux types flottant en C : float et double.
 - *float* (4 octets) : $\pm[1.17549435 \times 10^{-38} \dots 3.40282347 \times 10^{38}]$
 - *double* (8 octets) : $\pm[2.2250738585072014 \times 10^{-308} \dots 1.7976931348623158 \times 10^{308}]$

Les Variables (4)

- **Affectation d'une variable :**
 - L'affectation est l'opération qui permet de donner une valeur à une variable.
La notation algorithmique est la suivante :
nomVariable <- *valeur* ;
Le symbole <- est lu 'reçoit' ou 'prend pour valeur'.
- Notation algorithmique :
- i* <- 3
i <- *i* + 1
- Notation C :
- i* = 3 ; /* Attention ! */
i = *i* + 1 ; /* ce n'est pas l'égalité mathématique */

Les constantes

- Un algorithme dépend aussi des constantes...



- ✓ Définir et caractériser une constante

Les Constantes

- L'intérêt d'une constante est d'offrir plus de lisibilité et plus de souplesse
- Une constante doit toujours recevoir une valeur dès sa déclaration.
Notation algorithmique : *PI* <- 3.1416 : réel
Notation C : *const float pi = 3.1416 ;*
- Utilisation de *#define* en langage C
 - *#define* n'est pas traitée par le compilateur mais par le préprocesseur qui fait bêtement du remplacement de texte.
#define PI 3.1416

Les Opérateurs et les Expressions

- Les opérateurs permettent d'effectuer des traitements sur les données



- ✓ Utiliser les différents types d'opérateurs

Les Opérateurs et les Expressions (1)

- Deux types d'opérateurs :

- Les opérateurs unaires :

le signe - pour l'opposé d'un nombre

l'opérateur de négation logique NON (Notation C : !)

De plus en C :

L'incréméntation (++) pré (++a) ou post (a++)

La décrémentation (--) pré (--a) ou post (a--)

- Les opérateurs binaires

Les Opérateurs et les Expressions (2)

□ Opérateurs binaires :

■ Les opérateurs arithmétiques :

l'addition (+), la soustraction (-), la multiplication (*), la division (/), le modulo (%).

■ Les opérateurs de comparaison :

égalité (==), supérieur (>), inférieur (<), inférieur ou égal (<=), supérieur ou égal (>=), différent (!=)

■ Les opérateurs logiques :

ET, OU (Notation C : &&, ||)

Les tables de vérités associées à ces opérateurs sont celles de l'algèbre de Boole.

Les Opérateurs et les Expressions (3)

□ L'ordre de priorité pour les différents types d'opérateurs est le suivant (du + prioritaire au - prioritaire) :

- opérateurs arithmétiques : (* ; / ; %) ; (+, -)
- opérateurs logiques : ET ; OU

□ Par exemple :

L'expression :

((3>4) ET (8==8)) OU (0!=4)) donne la valeur VRAI

alors que l'expression :

((3>4) ET ((8==8) OU (0!=4)) donne la valeur FAUX.

Les Opérateurs et les Expressions (4)

□ Opérateurs sur les bits disponibles avec le langage C :

- et binaire & $a \& b$
- ou binaire | $a | b$
- ou exclusif binaire ^ $a \wedge b$
- décalage gauche << $a \ll b$
- décalage droit >> $a \gg b$
- complément à 1 ~ $\sim a$ (négation bit à bit)

Les Entrées / Sorties

□ Communiquer avec l'environnement...



✓ Effectuer des Entrées / Sorties

Les Entrées/Sorties (1)

- La lecture d'une donnée au clavier se fait via la fonction :

Lire(nomVariable)

Notation C : utiliser la fonction *scanf()*

- L'écriture d'une donnée à l'écran s'effectue grâce à la fonction :

Ecrire(texte, nomVariable)

L'argument texte doit être mis entre guillemets.

Notation C : utiliser la fonction *printf()*

Les Entrées/Sorties (2)

- Exemples :

- *V ← 10*

Ecrire ("V vaut ", V, ".")

Ecrire("Donnez la valeur de X")

Lire(X)

La Séquence d'instructions

- Déroulement en séquence ... Sans évènement



✓ Ecrire une séquence d'instructions

La Séquence

Début

instruction1

instruction2

...

instruction(n-1)

instruction(n)

Fin

En notation C : remplacer Début et Fin par { et }

La Sélection

- Un choix simple ou multiple...



✓ Utiliser les structures de contrôle permettant la sélection

La Sélection (1)

- Une condition est une expression écrite entre parenthèses à valeur booléenne : VRAI ou FAUX.

Si (condition) Alors

action1

Sinon

action2

En notation C :

if (condition)

action1

else

action2

'Sinon' n'est pas obligatoire
Si condition Alors
action

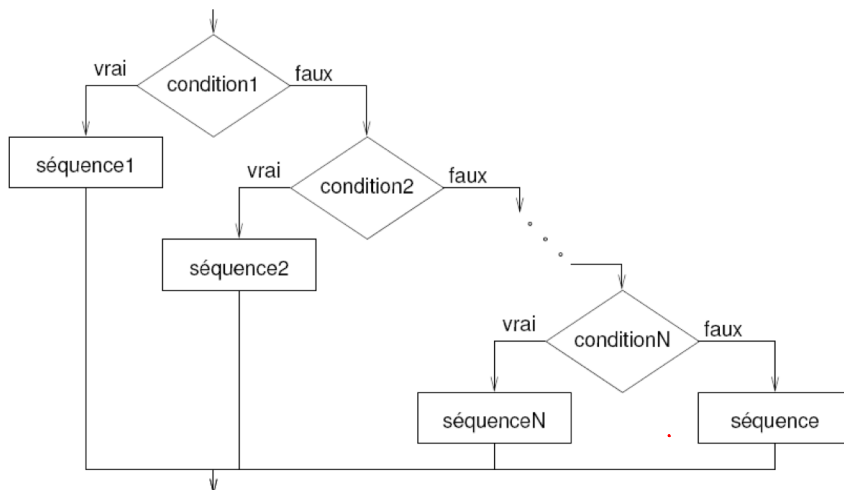
La Sélection (2)

□ Exercice :

- Reprendre l'algorithme de calcul de périmètre d'un cercle en effectuant un test de contrôle sur la saisie :

L3C2		fx =SI(Rayon>0; 2*PI()*Rayon; "Rayon Non Valide")				
	1	2	3	4	5	6
1						
2	Rayon =	-10				
3	Périmètre =	Rayon Non Valide				
4						

Tests imbriqués (1)



Tests imbriqués (2)

Si (condition_1) Alors
 Bloc d'instructions 1
Sinon Si (condition_2) Alors
 Bloc d'instructions 2
Sinon Si (condition_n) Alors
 Bloc d'instructions n
Sinon
 Bloc d'instructions final

Choix Multiples (1)

Selon expression Dans
 choix1 :
 Bloc d'instructions 1
 choix2 :
 Bloc d'instructions 2
 ...
 choixn:
 Bloc d'instructions N
Sinon
 Bloc d'instructions Défaut

Choix Multiples (2)

□ Notation en C :

```
switch (Variable) {  
  case Valeur1 :  
    Liste d'instructions;  
    break;  
  case Valeur2 :  
    Liste d'instructions;  
    break;  
  case Valeurs... :  
    Liste d'instructions;  
    break;  
  default:  
    Liste d'instructions;  
}
```

La Répétition

□ Un choix simple ou multiple...



✓ Utiliser les structures de contrôle permettant la répétition

La Répétition (1)

*TantQue (condition) Faire
action*

En notation C :

*while (condition)
action*

□ Remarque :

- La condition est évaluée avant de rentrer dans la boucle.
- la valeur de condition *doit être modifiée par le déroulement de l'action, sinon la répétition sera infinie.*

La Répétition (2)

□ Exemple :

- Saisir un rayon positif

Ecrire ("Rentrer le rayon:")

Lire (Rayon);

TantQue (Rayon \leq 0) Faire

Début

Ecrire ("Rentrer le rayon (positif) :")

Lire (Rayon)

Fin

La Répétition (3)

Faire

action

TantQue (condition)

En notation C :

do

action

while (condition);

□ Remarque :

- La condition est évaluée à la fin de l'itération de la boucle.
- la valeur de condition *doit être modifiée par le déroulement de l'action, sinon la répétition sera infinie.*

La Répétition (4)

Pour *cpt* de *val_init* à *val_fin* [ParPasDe *n*]

action

En notation C :

for (*exp_init*; *exp_cond*; *exp_iteration*)

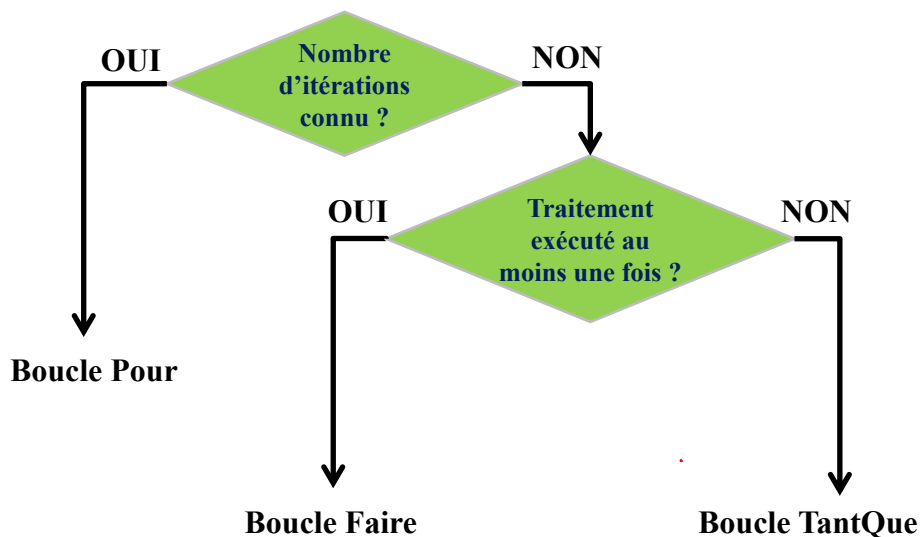
action

- *exp_init* est évaluée au début
- *exp_cond* est la condition pour continuer les itérations
- *exp_iteration* s'exécute à la fin de l'itération

La Répétition (5)

- Le langage C offre deux instructions :
 - *break* provoque la fin prématurée de la boucle while, do ou for qui contient directement le break ;
 - *continue* provoque le rebouclage immédiat, comme si on venait d'exécuter la dernière instruction du corps de la boucle.

La Répétition (6)



Les Structures de données composées

- Une structure de données composée permet de manipuler des données au sein d'une collection...



- ✓ Définir et comparer les différentes structures de données composées

Les tableaux (1)

- On appelle *tableau* une variable composée de données de même type, stockée de manière contiguë en mémoire (les unes à la suite des autres).
- On accède à un élément particulier d'un tableau grâce à son indice.
- Lorsque le tableau est composé de données de type simple, on parle de *tableau monodimensionnel* (ou *vecteur*)
Lorsque celui-ci contient lui-même d'autres tableaux on parle alors de *tableaux multidimensionnels* (aussi *matrice* ou *table*)

Les tableaux (2)

- Notation en algorithmique :

T1 : Tableau [Type_Indice] De Type_Élément;

- Notation en C :

type Nom_du_tableau [Nombre d'éléments]

- Exemples

Déclaration algorithmique : *notes : tableau[1..10] de réels*

notes : tableau[1..34] [1..3] de réels

Déclaration C : *float notes[10];*

float notes[34][3];

Les enregistrements (1)

- Contrairement aux tableaux qui sont des structures de données dont tous les éléments sont de même type, les enregistrements sont des structures de données dont **les éléments peuvent être de type différent et qui se rapportent à la même entité.**

- Les objets contenus dans l'enregistrement sont appelés champs de l'enregistrement.

- On accède à un champ grâce à l'opérateur '.'

nom_enregistrement . nom_champ

représente la valeur mémorisée dans le champ de l'enregistrement

Les enregistrements (2)

Type

```
nom_type = enregistrement
          nom_champ1: type_champ1
          ...
          nom_champn: type_champn
finenreg
```

Exemple:

Type

```
tpersonne = enregistrement
           nom : chaîne
           prénom : chaîne
           âge : entier
finenreg
```

Variable

```
salarié : tpersonne
```

Notation en langage C :

```
struct Nom_Structure {
    type_champ1 Nom_Champ1;
    type_champ2 Nom_Champ2;
    type_champ3 Nom_Champ3;
    ...
};
```