Algorithme

Dl Visual studio community 2015

Visual B

Visual c++

### C

Free Pascale (FPC)

Réf Int

**Intro à l’algo et à la programmation**

<http://perso.citi.insa-lyon.fr/afraboul/imsi/algo-imsi-4.pdf>

Cours et exo de Cormen, Leiserson, Rivest

Votre passeport informatique pour la programmation de Bruno Warin (ed Ellipses)

Procedure de réalisation d’un prog informatique :

L’écriture algorithmique est une phase intermédiaire et indispensable pour réaliser un programme.

La qualité du développement final dépend aussi de cette phase cruciale.

La suppression de cette phase contrairement à une idée courante de « gain de temps » qui en découlerait, est généralement la cause du retard de livraison finale du produit logiciel, et parfois de l’accroissement de la difficulté de résolution d’un programme complexe.

Notion de variable

Vous connaissez déjà la notion de variable en mathématiques.

Elle diffère cpdt de la notion informatique

La place étant limitée dans l’ordi, il convient de « dimensionner » la taille des « boites ». En prenant des grdes « boites », nous sommes sûr de pouvoir y stocker ce que l’ont veut mais le nombre de boites que l’ont peut stocker diminue. Si on prend des « boites » + petite on augmente la cap de stockage mais a une « variable » sera pt etre trop grd pour y etre rangée.

Il est donc évident que pour manip une variable elle doit d’abord être déf.

Les opé élém possibles sur les variables sont :

-Déclaration

-Affectation

-Utilisation

-Ré-affectation

Déclaration

Nom\_de\_la\_variable : <TYPE>

Nom de la variable = Nom attribué à la variable déclarée.

TYPE = le type est tjrs précédé du signe « : »

Nous verrons par la suite le type de variable dans le détail. Vous avez déjà vu en math les ensemble R, R+, N, N+

Les types élém possible sont :

-les entiers

-les réels (positif et négatif)

-les caractères

-les chaînes de caractères

-les booléens (vrai ou faux)

Déclaration

Nom de la variable = nom attribué à la variable déclarée.

Type = le type est tjrs précédé du signe « : »

(Nom de la variable) note\_ sur\_20 : ENTIER

Autre exemple

Nom des variables = nom attribué à la variable déclarée.

Type = le type est tjrs précédé du signe « : »

Nom\_var1, var2, var3 : < TYPE >

Nom\_de\_la\_variable ← valeur

← = symbole d’affectation

Note\_sur\_20 ← 15

Nom\_de\_la\_variable ← nom\_varible2

ATTENTION : Les 2 variables doivent être du même type.

Réaffectation

Note\_sur\_100 ← note\_sur\_20\*5

Note\_sur\_100 ← note\_sur\_100+5

Exercice

Var : a,b c : ENTIER

DEBUT

a←…

b←…

c←a

a←b

b←c

FIN

Une variable du précédent : on dispose de trois variables a,b et c. Ecrivez un algo transférant à b la valeur a, à c la valeur de b et à a la valeur de c (tjrs quel que soient les contenus préalables de ces variables).

Var : a,b,c x=entier

DEBUT

a←….

b←….

c←….

x←c

c←b

b←a

a←x

FIN

Notion de constante

-les Variables : VAR

-Déclaration

-Affectation

-Utilisation

-Ré-affectation

-Les Constantes : CONST

-Initialisation (avec déclaration)

-Utilisation

Déclaration

Nom\_de\_la\_constante ← valeur

Valeur= valeur définitive.

Exemple :

Π(PI) ← 3.14

Pourquoi utiliser une constante plutôt qu’une variable ?

-Opti

-Fiabilité

-Souplesse

-Facilité de relecture

-les données manipulées par les instructions sont :

- des variables proprement dites

-des constantes

-des valeurs littérales (« Zombie U », 45, VRAI)

-des expressions complexes (combinaison de variables, constantes et valeurs littérales avec des opérateurs.)

**Primitives**

- une instruction est un ordre élém – au sens algorithmique, que peut exécuter un programme.

Deux instructions d’entrée-sortie :

- Lire : lecture de la frappe (au clavier)

-ECRIRE : affichage (à l’écran)

Exemple d’algo

ALGORITHME : nom\_algorithme

//BUT : cet algorithme effectue…

//ENTREE : une liste de M noms

//Sortie : une liste triée par ordre alphabétique

//CONSTANTES : les constantes

//l’initialisation est **obligatoire**

// Au moment de leur déclaration

CONST : Pi 🡨 3.1416 : REEL

Nom 🡨 « Bonjour » : CHAINE

// Variables : les variables (au sens strict)

**VAR**: prix, quantité : REEL

Prenom : CHAINE

DEBUT

// INSTRUCTIONS

FIN

Autre exemple

ALGORITHME : calcul1

//BUT : Cet algo effectue la somme de trois entiers

//ENTREE : Trois entiers saisis par l’utilisateur

//Sortie : la somme

Var : nb1, nb2, nb3, somme :ENTIER

DEBUT

ECRIRE « VEUILLEZ ENTRE 3 ENTIERS »

LIRE nb1, nb2 , nb3

Somme 🡨 nb1+nb2+nb3

ALGORITHME : calcul2

//BUT : Réaliser somme, produit et quotient de ces valeurs.

//ENTREE : Deux entier saisi par l’utilisateur

//SORTIE : la somme, le produit et le quotient de ces valeurs.

Var : nb1, nb2, somme, produit ENTIER

Quotient : REEL

DEBUT

LIRE nb1, nb2,

Somme 🡨nb1 + nb2

Produit 🡨 nb1 \* nb2

Quotient 🡨 nb1 / nb2

ECRIRE somme

ECRIRE produit

ECRIRE quotient

FIN

SI nb2 vaut 0 pour le quotient sa bug

**ATTENTION**

Le développeur doit s’assurer qu’aucune « aberration » ne peut se produire

Partie 1 Module 2

Les Types élém possible sont :

- les entier

-les réels

-les caractères

-les chaines de caractères

-les booléens

ENTIER

Opé possible Opérateurs associe

Symboles, mot clé corres

Addition +

Soustraction -

Multiplication \*

Division DIV (pour la div entiere)

Exposant

Modulo

Comparaisons

Les Réels

Opération possible

Addition

Soustraction

Multiplication

Division

Exposant

Structure algorithmique

Catégories

Séquences

Rupture de séquences :

Rupture conditionnelle (et non réplique)

SI

CAS

Rupture répétitive :

POUR

TANTQUE

REPETER

SI. ALORS.. FINSI

SI..ALORS.SINON..FINSI

CAS…PARMI…

CAS…PARMI…PARDEFAUT

Exemple SI

SI <condition> ALORS

<instruction>

[SINON <instruction>]

FIN

CAS Parmi

CAS <variable> PARMI

{CAS{value[,]} : <instructions>}

[PARDEFAUT : <instructions>]

FINCARPARMI