Règles de programmation

Par selkis

Date de publication : 5 mars 2005

Dernière mise à jour : 7 mars 2013

Dans le cours qui va suivre, nous allons utiliser un pseudo-langage, comportant toutes les structures de base d'un langage de programmation.

I - Introduction

Un programme doit être le plus lisible possible, de manière à ce que n'importe qui d'autre que l'auteur soit capable de comprendre ce que fait le programme rien qu'en le lisant. Pour cela, il faut suivre quelques règles :

1. Le nom des variables doit être significatif, c'est-à-dire qu'elles indiquent clairement à quoi elles servent.
2. Un algorithme ne soit pas être trop long, sinon il faut le découper en fonctions et procédures.
3. Les structures de contrôle doivent être indentées, c'est-à-dire, par exemple, que les instructions qui suivent le ALORS doivent toutes être alignées et décalées d'une tabulation par rapport au SI. Il en est de même pour les répétitives.
4. À chaque imbrication d'une structure de contrôle, on décale d'une tabulation.

EXEMPLE MAL ECRIT

```
ENTIER a,b,c,i,j,k
TABLEAU CAR tab(10)(20)

i <--1 ; a<--0 ; b<--0
FAIRE
SI (tab(i,j)<>'')
ALORS
c <-- 0 ; a <--a + 1
j <-- 1
TANTQUE ((j <20) ET (c <> 0)) FAIRE
SI tab(i,j) = 'X' ALORS c <--0 FSI
j <--j + 1
FTQ FSI
SI (c=0) ALORS b <--b +1 FSI
i <-- i +1
JUSQUA (i > 10);
```

EXEMPLE MIEUX ECRIT

```
ENTIER i,j;
ENTIER nbmots, nbavecx;
BOOLEEN trouve;
TABLEAU CAR mts(10) (20);

i <--1 ; nbmots <--0 ; nbavecx <--0 ;
FAIRE
  SI (mots(i,j) <> '') ALORS
       trouve <-- FAUX ;
       nbmots <-- nbmots +1 ;
       j <--1 ;
    TANTQUE ((j<20) ET (NON trouve))
      FAIRE
        SI (mots(i,j)='X' ALORS
          trouve <-- VRAI
        FSI
     FTQ
      SI trouve ALORS nbavecx <-- nbavecx + 1 ; FSI;
  FSI
i <-- i +1 ;
JUSQUA (i >10)
```

II - Caractères utilisés

1. Les majuscules A....Z
2. Les minuscules a....z
3. Les chiffres 0.....9
4. Les signes = < > ' () [] * + - / , ; : / . Espace

Aucune distinction entre les majuscules et les minuscules n'est faite

III - Mots réservés et symboles

Les mots réserves sont les mots prédéfinis du langage algorithmique.

Ces mots ne pourront pas être employés pour définir d'autres objets du langage.

alors	autrecas	booleen	caractère
choix	constantes	créer	de
début	détruire	div	écrire
enregistrement	entier	entrée	et
faire	faux	fermer	fichier
fin	finchoix	finenregistrement	finfichier
finsi	fintantque	fonction	indexé
jusqu'à	lire	mod	non
null	ou	ouvir	pointeur
positionner	procèdure	programme	quelconque
réel	répéter	retourner	si
sinon	sortie	sur	tableau
tantque	types	variables	vrai

Les symboles prédéfinis sont des compositions de signes qui ont un sens particulier dans le langage algorithmique.

:=	>=	//
< >	<=	->

IV - Identificateurs

Les identificateurs sont les noms qui seront donnés aux différents objets déclarés dans un algorithme. Ils peuvent être, une suite de chiffres, de lettres (minuscules ou majuscules) et de soulignés, de longueur quelconque, commençant forcément par une lettre.

An98	**compte_dif**	**SDSDSD**	**s2011**	**[t_v_a]**

On ne fait pas la distinction entre les majuscules et les minuscules : nbr_table, NBR_Table, Nbr_Table, représente le même identificateur.

V - Fonctions et Procédures

1. On doit toujours être capable de donner un nom significatif à une procédure ou à une fonction.
2. Le nombre de paramètres ne doit pas être trop grand (en générale <5) car celui nuit à la lisibilité du programme.
3. Une procédure ou une fonction doit être la plus générale possible de manière à pouvoir être réutilisée dans d'autres circonstances.
4. Si le but d'une procédure est de calculer une valeur simole, il est préférable d'en faire une fonction.
5. Il est souvent plus clair d'écrire une fonction booléenne plutôt qu'une condition complexe.