LES LANGAGES EVOLUES

Ce chapitre présente une manière de programmer beaucoup plus aisée, grâce aux langages évolués.

INTRODUCTION

Nous avons déjà réalisé quelques programmes avec notre ordinateur : des dessins, des animations, de l'affichage de texte … Aussi la moindre tâche nous a-t-elle demandé un travail appréciable d'analyse et de construction pour arriver à nos fins. Par exemple, pour afficher le texte « Bonjour, Monsieur… » à l'écran, plutôt que d'écrire des dizaines de code assembleur comme nous l'avons fait, il serait plus aisé d'écrire quelque part :

AFFICHE « Bonjour, Monsieur… »

pour que cela se réalise, AFFICHE étant une commande qui demande à l'ordinateur d'afficher à l'écran le texte qui suit. De même, pour tracer une ligne entre deux points, il serait intéressant d'avoir une commande du type :

TRACE 12,23,345,85

où 12 et 23 d'une part, et 345 et 85 d'autre part sont les coordonnées du point de départ et d'arrivée du trait.

C'est ce que permettent deux types de programmes : les interpréteurs et les compilateurs. Tous deux permettent à l'utilisateur d'écrire son programme dans un langage directement compréhensible par l'homme car offrant un niveau de synthèse de son niveau. Ils se chargent de traduire ses instructions en une série de codes assembleurs pour que l'ordinateur soit capable de les exécuter. La différence entre les deux est que l'interpréteur traduit les instructions au fur et à mesure où il les découvre, alors que le compilateur traduit tout le programme dans un premier temps, puis l'exécute.
Quelques principes généralement appliqués :

- le programme comprend une instruction par ligne
- une ligne, donc une instruction, peut être repérée par une étiquette, l'équivalent de l'adresse de l'assembleur. Celle-ci n'est pas obligatoire, mais en mettre une permet de réaliser un branchement depuis une autre partie du programme (équivalent du JP de l'assembleur).
- il est possible de définir des variables, c'est-à-dire des noms auxquels on peut affecter une valeur, qu'on peut modifier par la suite (un peut comme des registres dont on aurait le loisir de définir le nom).
- chaque instruction est structurée par une syntaxe bien précise, définissant la manière de la rédiger. Par exemple, l'instruction AFFICHE doit être suivie par un blanc, puis la valeur à afficher. Si celle-ci est du texte, celui-ci doit être écrit entre guillemets. Sinon, c'est qu'il s'agit d'une valeur numérique, soit directe soit indirecte. Exemple :
    - texte : AFFICHE « A » Affiche le mot A à l'écran
    - directe : AFFICHE 10 Affiche le nombre 10 à l'écran
    - indirecte : Nombre = 10 Définit le nom Nombre comme portant la valeur 10
      AFFICHE Nombre Affiche la valeur prise par Nombre, c'est-à-dire 10
- cela nécessite de la part de l'utilisateur une grande rigueur dans le respect de la syntaxe du langage utilisé, sinon l'interpréteur ou le compilateur ne comprendra pas l'instruction et génèrera un code d'erreur.

Nous ne détaillerons pas le fonctionnement intime des interpréteurs et compilateurs (ce sujet demanderait un volume entier à lui seul), mais en voici les principales étapes :
contrôle de la syntaxe : il s'agit de vérifier que les règles de rédaction des instructions ont bien été respectées. Cela se fait par rapport à une bibliothèque de références.###

- reconnaissance des instruction, afin de lancer les bonnes interprétations
- création des variables
- gestion des liens
- génération du code machine. #vérifier dans le bouquin sur les compilateurs#

LES INSTRUCTIONS

Quelques exemples d'instructions (en majuscule) :

AFFICHE « *texte* » AFFICHE *expression numérique*	Affiche le texte Affiche l'expression (qui peut être : Nombre/34)
ALLER A *étiquette*	réalise un branchement vers l'étiquette
SI *condition* ALORS *instruction*	réalise ou pas une instruction selon la réalisation d'une condition
DEMANDE *variable*	demande à l'opérateur de saisir une valeur, qui sera mémorisée dans *variable*
TANT QUE *condition* FAIRE *instruction*	exécute l'*instruction* tant que la *condition* n'est pas réalisée
FAIRE *instruction* JUSQU'A *condition*	exécute l'*instruction* jusqu'à ce que *condition* soit réalisée
DE *variable* = *début* JUSQU'A *fin* FAIRE *instruction*	exécute l'*instruction* avec *variable* prenant la valeur *début*, puis avec *variable* prenant la valeur *début*+1, puis avec *variable* prenant la valeur *début*+2 et ainsi de suite jusqu'à *variable* = *fin*.
DEBUT instruction 1 instruction 2 … instruction n FIN	définit un bloc d'intructions comme une instruction
APPELLE *étiquette* FIN	appelle un sous-programme situé à *étiquette* retour du sous-programme dans le programme appelant
DESSINE *x*,*y*,*couleur*	Dessine un point de couleur *couleur* en *x*,*y* sur l'écran
TRACE x_1,y_1,x_2,y_2,*couleur*	Trace une droite depuis x_1,y_1 jusqu'à x_2,y_2 de couleur *couleur*
RECTANGLE x_1,y_1,x_2,y_2,*couleur*	Trace un rectangle de diagonale x_1,y_1 jusqu'à x_2,y_2 et de couleur *couleur*
ELLIPSE *x*,*y*,r_x,r_y,*couleur*	Trace une ellipse de centre *x*,*y*, de rayon horizontal r_x, de rayon vertical r_y et de couleur *couleur*. Si $r_x = r_y$, alors c'est un cercle.
PROCEDURE *Nom* (*arguments*) DEBUT instructions… FIN	Crée une nouvelle instruction appelée *Nom*. Exemple : PROCEDURE CERCLE (x,y,rayon) DEBUT ELLIPSE (x,y,rayon,rayon) FIN

etc…

LES FONCTIONS

Un langage comprend également des fonctions. Une fonction est un nom qui prend une valeur dépendante de ses arguments (elle peut ne pas avoir d'arguments, dans ce cas les parenthèses sont vides). Par exemple :

SIN (PI/2) = 1	sinus
ABS (4) = 4	valeur absolue
ABS (-4) = 4	valeur absolue
ENTIER (5,65) = 5	partie entière

ALEATOIRE ( )= 0,546857 ou 0,214578 ou …	valeur aléatoire entre 0 compris et 1 non compris
ASCII (32) = « »	caractère ayant le code ASCII spécifié
CODE (« ») = 32	code ASCII du caractère spécifié
TOUCHE_PRESSEE ( ) = « k » ou « p » ou « g »…	touche pressée au moment de l'exécution de la fonction
TOUCHE_PRESSEE ( ) = «»	si aucune touche n'est pressée au moment de l'exécution de la fonction
FONCTION *Nom* (arguments) DEBUT instructions… FIN	Crée une nouvelle fonction appelée *Nom*. Exemple : FONCTION MULTIPLIE_PAR_10 (n) DEBUT MULTIPLIE_PAR_10 = n × 10 FIN

UTILISATION