4. Des humains et des langages

Bonjour. Bienvenue dans cette dernière séquence sur la programmation. Je vais parler un peu plus des langages de programmation, des types de langage de programmation, et évidemment des humains qui programment.

Il y a beaucoup, beaucoup, beaucoup de langages de programmation, qui ont des caractéristiques différentes. Ce qu'il faut retenir, c'est qu'on peut toujours faire tout dans tous les langages de programmation. Par contre, ça ne sera pas toujours aussi facile, mais vous pouvez utiliser n'importe quel langage de programmation pour écrire n'importe quel algorithme. On peut classer les langages de programmation par type. On appelle ça des paradigmes. Je vais vous citer un certain nombre de paradigmes de programmation. Le plus usuel, c'est le paradigme impératif qui est le plus proche du processeur, on donne des ordres successivement, on fait des boucles, ce genre de choses. Les exemples les plus classiques sont Fortran et C. Il y a les langages fonctionnels où on considère les fonctions comme des objets à part entière qu'on peut passer à d'autres fonctions. C'est une vision un peu plus mathématique de la programmation. Les exemples de tel langage sont Haskell et OCaml. La programmation objet est une façon de faire de la programmation un peu plus générique, où effectivement on va pouvoir partager un certain nombre de fonctions entre des objets différents. La programmation par contrainte, un exemple est Prolog, est plus une programmation logique. Imaginez un calcul d'emploi du temps, on a un certain nombre de contraintes, tel professeur n'est là que lundi matin, telle salle ne peut faire que de l'informatique, etc. Avec ça, on met toutes ces contraintes et on obtient idéalement un emploi du temps correct pour tous les professeurs et toutes les salles. La programmation événementielle est plus utilisée pour les interfaces graphiques ou la robotique. On n'a pas le programme qui se déroule point par point, mais on attend un certain nombre de stimulus de l'environnement. Imaginez un robot, on va toucher le robot, on va appuyer sur un bouton et à ce moment-là, le programme doit réagir à ces stimulus.

Il faut aussi comprendre que la plupart des langages de programmation sont multiparadigmes. En C++, vous pouvez faire de l'impératif, de l'objet et d'autres choses. En OCaml, vous pouvez faire de l'impératif, du fonctionnel et de l'objet, en Python également. En Java, on fait beaucoup d'objet, mais on peut aussi faire de l'impératif et du fonctionnel. La plupart des langages qu'on utilise sont multiparadigmes et on peut les utiliser d'ailleurs d'une façon ou d'une autre.

Un petit exemple pour vous montrer les façons différentes de programmer. Imaginons qu'on ait un ensemble de valeurs entières et qu'on veuille ajouter un à toutes ces valeurs. En C, ce qu'on ferait naturellement, c'est qu'on choisirait un tableau d'une certaine taille, un int*, un pointeur vers un tableau d'entiers, et on le parcourrait séquentiellement. On aurait une boucle pour, où on va parcourir chaque case du tableau et sur chaque entier, on ajoute 1, successivement, à chaque entier. Par contre, ce qu'on ferait plutôt en OCaml, on choisirait une liste parce que c'est un langage plus fonctionnel, donc au lieu d'avoir un pointeur ou un tableau, on aurait un int list, une liste d'entier et on appliquerait un opérateur qui a un map, qui est quelque chose qui va appliquer une fonction sur l'ensemble des éléments d'une liste. Effectivement, cette fonction map prend une fonction qui à un entier associe un entier successeur. On ferait ce map qui prend comme entrée une fonction et cette liste, qui va appliquer successivement ce +1 à tous les éléments de cette liste.

Au final, choisir un langage de programmation, c'est toujours une question compliquée. Ça dépend de ce qu'on veut programmer. Ça dépend de ce qu'on a envie de faire, de ce qu'on a l'habitude de faire. Quel paradigme on a envie de faire, d'utiliser ? Quelles interfaces avec les autres développements parce que souvent notre programme n'est qu'un morceau de quelque chose qui est plus gros, dans ce cas-là, pour interfacer souvent, il faut utiliser le même langage. Qu'est-ce qu'il y a comme bibliothèques support ? Par exemple, si je veux faire une interface graphique, j'ai envie d'utiliser des bibliothèques existantes. Effectivement, quelle est la bibliothèque que j'ai envie d'utiliser ? Qu'est-ce que je connais du langage ? On ira beaucoup plus vite sur un langage qu'on connaît. Est-ce que je peux obtenir de l'aide ou du tutorat ? Si mon voisin est très fort dans un langage, je vais peut-être utiliser celui-là parce que je sais que je pourrais lui poser toutes les questions dont j'ai besoin. Au final, choisir un langage, c'est toujours compliqué, mais on choisit le langage qu'on veut. Avec ce langage, on peut faire toujours tout ce qu'on veut.

En conclusion sur cette partie sur la programmation, on a commencé par le langage machine, le plus proche du processeur, qu'on n'a pas envie d'écrire. C'est pour ça qu'on utilise des langages de plus haut niveau et qu'on utilise un compilateur pour écrire le langage machine. Ce compilateur va, en plus, nous aider à trouver un certain nombre d'erreurs de syntaxe sur le programme. Par contre, il ne fait pas tout. Il n'est pas capable de nous dire que le programme fait ce qu'on veut. C'est pour ça qu'il faut tester le programme. En conclusion, vous n'avez plus qu'à programmer parce que la meilleure façon de s'entraîner, c'est de programmer effectivement soi-même. Bon courage !Il peut aussi y avoir une erreur de communication entre 2 programmeurs, parce que les programmes actuels ne sont pas écrits par une seule personne, ils sont écrits par des milliers de personnes qui doivent interagir, ils doivent comprendre ce que fait le morceau de programme des autres personnes.

Pour vous montrer qu'effectivement les programmes sont très gros, je vous ai montré un petit exemple avec un dessin qui représente le nombre de lignes de code d'un certain nombre de programmes. Vous pouvez retrouver ce dessin en meilleure résolution à cette adresse web, et donc je vais vous en expliciter quelques-uns. Un pacemaker c'est à peu près 100 000 lignes de code, donc vous voyez c'est déjà difficile à écrire par une seule personne et ce n'est qu'un petit programme par rapport à ce dont je vais vous parler. Après, le télescope spatial Hubble, à l'intérieur il y a 2 millions de lignes de code, donc c'est déjà beaucoup. Un autre exemple de taille supérieure, c'est à peu près 5 millions de lignes de code, ce qui correspond aux serveurs de World of Warcraft, ou du code embarqué dans un Boeing 787. Alors encore plus gros, vous avez Facebook qui représente à peu près 60 millions de lignes de code. Donc vous imaginez bien que c'est très compliqué sur 60 millions de lignes de code, d'être sûr qu'il n'y a pas une erreur qui s'est insérée quelque part, et c'est pour ça que vos programmes de temps en temps, ils ne donnent pas leurs bons résultats, ou ils plantent.