Pourquoi Ada ?

1) Introduction

Qu'est ce qui me pousse à exprimer ma préférence pour le langage informatique Ada, pour le moins de la partager, voire de la recommander ?
Je pourrais simplement faire référence à de très bon plaidoyers (en anglais) existants sur Internet (http://www.adaic.org/whyada/index.html par exemple). Curieusement, je n'ai pas trouvé de réquisitoire contre (serait-ce de la mauvaise foi inconsciente ?) Bref, mon propos sera plus personnel. Mon adhésion à Ada est la conséquence d'une histoire propre qui commence avec les débuts de la micro-informatique dans les années 80 du siècle dernier. Bien que pas encore papi, la naissance de la micro-informatique a été concomitante avec mon passage à l'âge adulte avec ses espoirs et sa soif d'apprendre des sujets qui n'avait jamais existé auparavant.

2) Le Basic

Comme pour beaucoup à cette époque des années 80, mon premier apprentissage d'un langage informatique a été le langage Basic (créé en 1963). À l'époque très populaire, il est présent sur la plupart des tous premiers micro-ordinateurs l'Apple II bien sûr mais aussi le Sinclair ZX81, le Commodore VIC-20, le Texas TI99/4A, l'Atari 800XL entre autres pour ceux qui me sont passés entre les mains et même les calculatrices évoluées comme la Sharp PC-1500. Il comporte peu d'instructions, qui, interprétées, permettent une inter-activité immédiate avec l'utilisateur. Son initiation est très rapide pour quantité de personnes. Le magazine Hebdogiciel (https://fr.wikipedia.org/wiki/Hebdogiciel) publia de 1983 à 1987 des dizaines de programmes en Basic envoyés par les lecteurs.

Néanmoins, il s'avérera très vite limité malgré l'apparition de plusieurs Basics "étendus". L'apprentissage de l'assembleur devint ainsi plus que nécessaire. S'il n'était pas aisé de programmer lisiblement en Basic (au sens de suffisamment lisible pour repérer rapidement - à la première ou au plus à la deuxième lecture - des erreurs de programmation parmi des dizaines de lignes de programmes), la vérité est que la structure des premiers Basics n'aidait pas. Les instructions de saut GOTO et les variables toutes globales déstructuraient le meilleur des programmes codé même avec la meilleur des volontés.
Il en a été tout autre en assembleur.

3) L'assembleur

Avec l'assembleur, une méthodologie s'imposait : une erreur était généralement fatale pour le fonctionnement du micro-ordinateur, même pas droit au mythique "Syntax error" rudimentaire du Basic mais salvateur.

La pratique de l'assembleur était alors le summum de la reconnaissance. Il fallait alors non seulement connaitre les instructions propres à chaque micro-processeur mais aussi le fonctionnement interne de son micro-ordinateur pour pouvoir utiliser les instructions correctement. Sa maitrise permettait de s'affranchir des limitations du Basic dont la première était la lenteur. Quelle surprise, quelle joie, d'exécuter un programme assembleur pour s'apercevoir que sa rapidité d'exécution est quasi instantané remettant radicalement en cause la manière de le déverminer et donc notre façon de programmer.

La littérature de l'époque était somme toute très confidentielle. Quelques livres et magazines dont Hebdogiciel livraient de précieuses informations techniques mais pas de méthodologie.

Paradoxalement, une méthodologie de programmation avait tout son sens avec le langage assembleur, même, elle est bien plus facile d'application qu'en Basic car l'assembleur est à l'évidence plus souple. Cette méthodologie assez personnelle pour ma part s'apparentait à la programmation structurée : localisation des variables, bibliothèque de fonctions avec protocole d'entrée / sortie... Néanmoins, c'était laborieux d'écrire des lignes et des lignes d'instructions pour faire des choses somme-toute simples et répétitives.

4) Le Pascal

Puis vint Turbo Pascal et avec lui la programmation structurée native au langage Pascal. Je l'ai abordé au moment de la transition importante entre les versions 3.0 et 4.0. Cette dernière apporta les unités de compilations séparées avec les sections interface et implémentation. Le manuel accompagnant le logiciel très complet reste un modèle du genre. Je lus les 640 pages d'une seule traite. Je découvre alors le typage des données, la récursivité, les pointeurs... pendant les vacances d'été. Je commence à écrire des programmes Pascal sur papier alors que je n'ai pas de PC pour faire exécuter mes programmes. À cette époque, ils sont encore hors de prix alors que TP est très abordable dans sa version pour l'éducation. Il n'y a pas de doute Pascal est un langage de programmation passionnant. Son succès est rapide dans l'éducation et dans le monde professionnel. Il va ensuite principalement s'enrichir de fonctionnalités orientées objets avec la version 5.5.

Par rapport au basic de l'époque, les avantages sont nombreux avec notamment : disparition de l'obligation du goto, disparition des numéros de lignes, apparition des sous programmes utilisateurs autonomes avec possibilité de récursivité, apparition des types utilisateurs, des pointeurs en mémoire.

5) L'Ada

A la fin de mes études, pour mon premier job, j'ai le plaisir de développer des programmes en Pascal. Je suis alors sélectionné pour une formation à Ada dans le cadre professionnel. Ada m'évoque juste un langage exigeant, une programmation plus rigoureuse voire sans doute complexe. L'aperçu, que j'en avais eu lors de mes études notamment la notion de rendez-vous, m'était passé complètement au-dessus.

Pendant cette formation très réputée chez Adalog, c'est une révélation. Ada a non seulement une filiation avec Pascal mais en plus il en comble les principales lacunes comme les types génériques, les surcharges de procédure, les définitions d'opérateurs, les tableaux non contraint... Je vois alors Ada comme un Pascal++ alors qu'Ada est beaucoup plus que cela. On ne programme plus en Ada comme on pouvait le faire en Pascal. Avec Ada il faut obligatoirement penser méthodologie avant toute écriture de code.

Professionnellement, les programmes Ada sont en version 83 mais personnellement avec l'arrivée d'Internet je prend connaissance de la version 95 avec ses grandes nouveautés l'héritage, le polymorphisme entre autres.

A travers ses différentes évolutions 2005, 2012 et maintenant 202x, Ada ne cesse de s'enrichir avec toujours une très grande cohérence.

Pascal Pignard, octobre 2010, septembre 2020.