1. Linguaggio C

Nel 1972 Dennis Ritchie progettava e realizzava la prima versione del linguaggio C presso i Bell Laboratories. Ritchie aveva ripreso e sviluppato molti dei costrutti sintattici del linguaggio BCPL, di Martin Richards, e del linguaggio B, di Ken Thompson, l'autore del sistema operativo UNIX. Successivamente gli stessi Ritchie e Thompson riscrissero in C il codice di UNIX.

Il C si distingueva dai suoi predecessori per il fatto di implementare una vasta gamma di tipi di dati - carattere, interi, numeri in virgola mobile, strutture - non originariamente previsti dagli altri due linguaggi. Da allora ad oggi non ha subito profonde trasformazioni: la sua sintassi è stata estesa, soprattutto in conseguenza della programmazione orientata agli oggetti (C++), ma nella sostanza e nello spirito il linguaggio è rimasto quello delle origini. Il C è un linguaggio di alto livello che possiede un insieme ristretto di costrutti di controllo e di parole chiave, ed un ricco insieme di operatori. Consente di programmare in modo modulare, per mezzo delle funzioni e delle macro, anche se non esiste una gerarchia di funzioni come, ad esempio, in Pascal. Pur essendo un linguaggio ad alto livello permette operazioni di basso livello tipiche del linguaggio macchina: si può, ad esempio, indirizzare la memoria in modo assoluto, funzionalità fondamentale per lo sviluppo di applicazioni di basso livello. E' un linguaggio apparentemente povero: non possiede istruzioni di entrata/uscita, ne istruzioni per operazioni matematiche. Ogni funzione diversa dai costrutti di controllo o dalle operazioni elementari sui tipi dati è affidata ad un insieme di librerie esterne. In questo modo, Ritchie riuscì a raggiungere due obiettivi: da una parte, mantenere compatto il linguaggio, dall'altra, poter estenderne le funzionalità semplicemente aggiungendo nuove librerie o ampliando quelle esistenti. E' stato talvolta definito come "il linguaggio di più basso livello tra i linguaggi di alto livello". Infatti, come abbiamo detto, nasce per lo sviluppo di sistemi operativi, quindi per software di basso livello, ma preservando la semplicità d'uso dei linguaggi della terza generazione.

Sono molti i fattori che hanno determinato la sua capillare diffusione. Il trampolino di lancio è stato il sistema operativo Unix. Il C ne ha seguito le sorti fin dall'inizio divenendo ben presto il linguaggio di programmazione preferito dalle università e dagli istituti di ricerca. Unix è stata la dimostrazione pratica della bontà e forza del linguaggio C. Il mondo dell'industria informatica lo ha notato ed oggi praticamente non esiste prodotto commerciale di larga diffusione - database, wordprocessor, foglio elettronico, browser etc. - che non sia scritto in C. Un'altro fattore che ha contribuito al successo del C è stato il personal computer. Quelle che erano funzioni di programmazione di sistema fino a qualche anno fa riservate a pochi specialisti oggi sono accessibili a tutti. Ad esempio, oggi è molto facile, anche per un programmatore C dilettante, pilotare direttamente l'hardware. In pratica il C sta sostituendo l'assembler. Esistono, infatti, dei compilatori talmente evoluti, che il codice assembler equivalente al C prodotto dal compilatore è talmente efficiente e compatto da risultare migliore di quello scritto anche da un buon programmatore assembler. I nuovi linguaggi che si sono presentati sulla scena dell'informatica, quali Java e Perl, devono molto al C, la cui conoscenza costituisce un ottimo punto di partenza per il loro apprendimento. L'enorme diffusione raggiunta e la sua efficienza, anche nel pilotare direttamente l'hardware, continuano a fare del C una scelta largamente condivisa anche per realizzare applicazioni Internet o parti di esse. Infine esistono anche motivi estetici. Per l'eleganza della sintassi e la compattezza dei costrutti, il C è una sfida permanente alle capacità intellettuali del progettista software, è anche una utilissima palestra per il giovane programmatore che con esso impara a risolvere una vasta classe di problemi, lo spinge a migliorare le proprie tecniche, e lo abitua a controllare le idiosincrasie della macchina senza dover ricorrere all'assembler.