История на езика С

Създаден 1972 в „AT&T Bell Laboratories“ от Dennis Ritchie с цел създаване на операционната система Unix.

Синтаксиса на С в последствие става основа на много съвременни модерни езици, като C++, D, Go, Rust, Java, JavaScript, Limbo, LPC, C#, Objective-C, Perl, PHP, Python, Verilog, Unix's C shell

Traditional C	1972	Dennis Ritchie
K&R C	1978	Brain Kernighan and Dennis Ritchie
ANSI C	1989	ANSI Committee
ANSI/ISO C	1990	ISO Committee
C99	1999	ISO/IEC Committee
C extensions to embedded processors	2006	ISO/IEC Committee
C11	2011	ISO/IEC Committee

Характеристики на езика

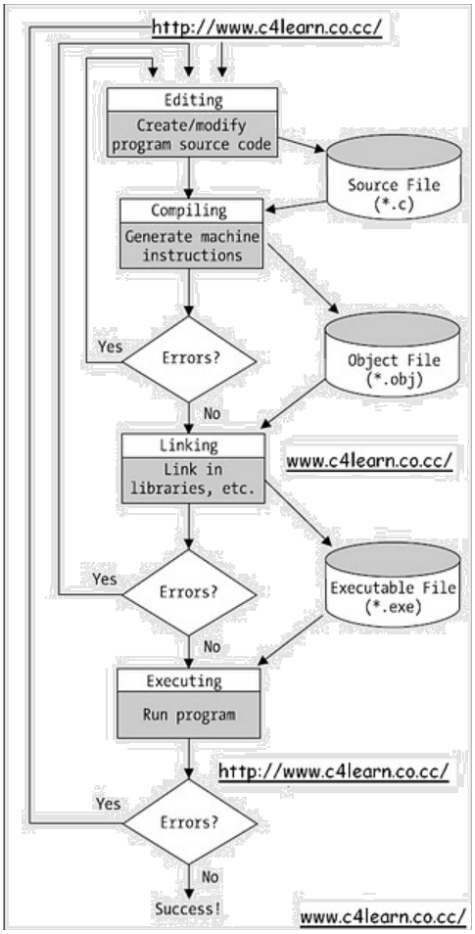
- Мощен език с възможности за достъп до ниско ниво – лесно включване на програми на асемблер, манипулация на битове, директен достъп до адреси от паметта чрез указатели, адресна аритметика.
- Преносимост базирана на пре-компилация на програмата.
- Възможност за създаване и манипулиране на най-разнообразни потребителски типове данни, широк набор от готови функции. Ограничен контрол върху типовете данни.
- Създаване на изключително ефективни програми от гледна точка на памет и бързодействие.
- Лесно създаване и комбиниране на програмни модули, които позволяват отделна настройка и комбинирането им за създаване на мощни програми.
- Частичен контрол върху действията на програмиста (ако той прави глупости, сигурно има нещо гениално предвид) – възможности за ефективни решения, но и за трудни за откриване грешки.

Приложения

Езикът продължава да е абсолютен лидер навсякъде където е необходимо програмиране на ниско ниво (паметта и бързината са от особено значение) – драйвери, вградени системи, операционни системи, езикови интерпретатори, управление на устройства.

Компилатори и интерпретатори

Етапи при създаване на програма (взето от http://www.c4learn.com)



Фази на компилатора

- **Лексичен анализ**: преобразува текста в лексеми – низ от символи във валиден езиков маркер (token). Например **int value = 100;** съдържа маркерите int (keyword), value (identifier), = (operator), 100 (constant) and ; (symbol)
- **Синтактичен анализ (parsing):** изгражда синтактично дърво от маркерите – проверява дали изразите са синтактически правилни.
- **Семантичен анализ:** проверява дали синтактично дърво е по правилата (граматиката) на езика (дефинирани идентификатори, присвояване между съвместими типове …)
- **Генериране на междинен код:** за абстрактна машина подходящ за оптимизация.
- **Оптимизация:** извършва се локална оптимизация – излишни променливи, излишни извиквания на функции, излишни цикли …
- **Генериране на код**