*Резюме:***С** (произнася се „си“ в оригинал на английски, защото е на латиница) е език за програмиране, разработен от Денис Ричи[1] в периода 1969 – 1973 г. Разработката на езика е тясно свързана с операционната система UNIX, чието ядро впоследствие бива пренаписано на езика C.

*Предназначение:*

С е език за програмиране от средно ниво с общо предназначение. Поради ниското ниво на абстракция, програмистите имат повече контрол върху хардуера и програмите написани на него обикновено работят по-бързо от тези, написани на езици от високо ниво. Затова C е подходящ за създаване както на операционни системи, така и на приложения. Преносимостта на кода съчетана с неговата ефективност прави езика доста популярен. Той получава изключително голямо разпространение, като за него съществуват компилатори за многобройни операционни системи и компютърни платформи. Много от днешните операционни системи са написани главно на C. Популярните езици C++ и Objective-C са разширения на езика С. Код, написан на С, може много лесно да се използва на различни платформи. За езика са разработени няколко стандарта - KNR, ANSI C иC99.

*Началото:*

Първоначално C сe разработва в AT&T Bell Labs между 1969 и 1973 г. Произходът на C е тясно свързано с разработването на операционната система UNIX, първоначалното написана на асемблер за компютъра PDP-7 от Ричи и Томпсън, които включват няколко идеи от техни колеги. За да обогати възможностите на системата, Томпсън създава системния език *B*, който представлява опростена версия на езика BCPL. По-късно UNIX се пренаписва за новата машина PDP-11, но заради някои недостатъци на езика B, поради които не може да се използва пълния потенциал на PDP-11, Денис Ричи решава да създаде езика C. Качествата на C като абстрактност и преносимост бързо го правят популярен и през 1973г. операционната система UNIX е пренаписана на този език което значително улеснява пренасянето ѝ на други машини.

*K&R C(Кернихан и Ричи):*

През 1978 г. Браян Кернихан и Денис Ричи написват книгата *The C Programming Language*. Тази книга (наричана за кратко *K&R*) се явява първата по-формална спецификация на езика и впоследствие версията на C описана в книгата започва да се нарича *K&R C*. Второто издание на книгата описва създаденият по-късно формален стандарт на езика ANSI C. Но дори и след излизането на *ANSI C*, много програмисти предпочитат да ограничават кода си в рамките на *K&R C* за максимална преносимост на кода, тъй като много от старите компилатори разбирали само *K&R C* а внимателно написан *K&R C* код може да бъде и съвместим с *ANSI C*.

*ANCI C & ISO C:*

Изключителната популярност на езика довежда до разработването на компилатори и разновидности на езика с различни разширения за множество компютри. За да подобри преносимостта на кода и за да осъвремени съществуващия стандарт, през 1983 г. Американският национален институт по стандартите (ANSI) сформира работна група имаща за цел да създаде формално описание на *C*. През 1989 г. стандартът е завършен и ратифициран под името *ANSI X3.159-1989 "Programming Language C"*. Версията на езика описана в документа често се нарича *ANSI C* или *C89*. През 1990 г. стандартът *ANSI C* се възприема и от Международната организация по стандартизация (ISO) под името ISO/IEC 9899:1990, накратко наричан C90. Тъй като съдържателно стандартите са едни и същи (различават се само по оформлението) *C89* и *C90* се отнася до един и същи език.

*Стандартът C99:*

C99 е нов стандарт на езика, утвърден през 1999 от ISO/IEC, добавящ някои важни и възможности, между които са:

* Inline функции
* Премахване ограничението за деклариране на променливи само в началото на блок
* Нови типове данни, включващи long long int, boolean и др.
* Променлива дължина на масивите
* Поддръжка на едноредови коментари //, като в някои други езици за програмиране
* Нови библиотечни функции
* Нови заглавни файлове, като stdbool.h и inttypes.h
* Подобрена поддръжка на IEEE floating point
* Добавяне на нови ключови думи, като restrict

*Характеристика:*

Програмите написани на C представляват съвкупност от файлове с изходен код. Чрез компилация всеки файл се превежда до машинно зависим обектен код за определена архитектура, а свързващ редактор обединява тези относително независими обектни файлове в цялостен изпълним файл. *C* е структурен език, използващ конструкции като условни преходи (if.. else), цикли (while, for) и абстрахиране на кода чрез функции изпълняващи относително самостоятелни задачи.

Някои от най-важните характеристики на езика са:

* Слабо типизиран (weak typing) - Позволява автоматично преобразуване между някои типове данни.
* Статично типизиран (static typing) - Типовете на данните се определят по време на компилирането.
* Позволява работа с паметта на ниско ниво — поддържа указатели към променливи, указатели към функции и адресна аритметика.
* Използва сравнително малък набор от ключови думи, като за сметка на това използва сравнително голямо количество оператори.
* Всеки израз връща стойност — дори присвояването, което прави възможно множествено присвояване от вида x=y=5. Стойностите обаче могат да бъдат игнорирани когато не са нужни.
* Параметрите към функциите винаги се предават по стойност. Така достъпът до външна променлива чрез параметър може да стане само индиректно чрез подаване на указател.
* Изобилстваща употреба на предпроцесор за създаване на макро дефиниции, включване на файлове и условно компилиране.

*Примерът Hello World!:*

Следва класическата примерна програма, извеждаща на екрана текста *Hello, World!* („Здравей, свят!“):

#include <stdio.h>

 int main(void)

 {

 printf("Hello, World!**\n**");

 return 0;

 }

Първият ред представлява директива (#include) към предпроцесора за включване на външен файл (stdio.h) който съдържа описанията на някои стандартни функции за вход и изход на данни. В случая е нужен за да може компилаторът да намери описанието на функцията за форматиран изход printf. Триъгълните скоби указват да се търси в каталога със стандартни заглавни файлове. Следващите редове представляват дефиниция на функция с име main като int пред името на функцията указва че тя връща стойност int (целочислен тип). Всяка програма трябва да има функция main и това е първата функция която се изпълнява при стартиране на приложението. В кръглите скоби се описват параметрите които функцията приема, но в случая тя не изисква никакви параметри и това се заявява с ключовата дума void. Във фигурните скоби е поместено тялото на функцията което ще се изпълни. Започва с извикване на функция от стандартната библиотека на *C* — printf, която в случая приема единствен параметър — указател към низ от символи. Двойните кавички създават анонимен низ (низова константа) от символите заградени в тях в паметта и връща указател към него. Нарича се анонимен защото не е присвоен на променлива и съответно по-късно няма да можем да се обърнем към него чрез такава променлива. Комбинацията от символи \n се нарича управляваща последователност, защото символите ѝ не се възприемат буквално от компилатора. Управляващи последователности се използват за вмъкване на символи които или не бихме могли директно да въведем или биха нарушили синтаксиса на програмата. В случая \n означава символ за нов ред; ако просто бяхме въвели истински, буквален символ за нов ред, компилаторът щеше да даде съобщение за грешка тъй като синтактичните правила изискват изразите да заемат един единствен ред. Символът **;** поставя края на израза който е основния градивен блок на програмите. Изразът return 0; прекратява изпълнението на функцията и връща целочислената стойност нула. Често функциите, освен ако нулата не е валидна резултантна стойност на задачата изпълнявана от функцията, връщат нула за да сигнализират че не е възникнал проблем при изпълнението им. Но това е различно за всяка функция и за смисъла на връщаните стойности програмистът трябва да се допита до документацията.