

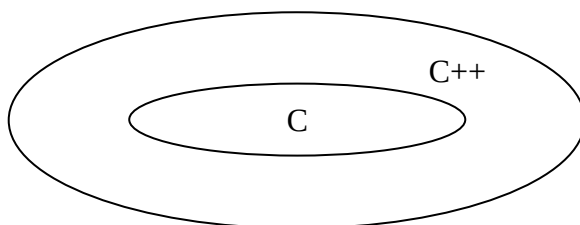
1.1 Om C/C++ och Python

70-
80-talet

C++ är en direkt utvidgning och vidareutveckling av programmeringsspråket C som 1972 utvecklades av Dennis Ritchie på Bell Laboratories med syftet att skapa ett språk för programmering av operativsystemet *Unix*. I den bemärkelsen är C en biprodukt av Unix. Därför finns många logiska paralleller mellan C och Unix. Idag är inte bara Unix utan även andra operativsystem inkl. Windows skrivna i C. Styrkan i C består av en kombination mellan enkelhet, strukturering och möjligheten att lätt kunna kommunicera med datorns hårdvara. C har bland de moderna språken den bästa förmågan att hantera och kontrollera hårdvaran, vilket favoriserar C som programspråk t.ex. för operativsystem, men även för inbyggda system. Den stora frihet som C erbjuder för hantering av bl.a. datorns primärminne med hjälp av *pekare*, kod som ger åtkomst till den fysiska adressen till data och på gott och ont tillåter manipulationer av minnesadresser.

Det var dansken Bjarne Stroustrup som la grunden till vidareutvecklingen av C. Under 70-talet hade man nämligen konstaterat att *procedural programming* (Algol, Pascal, C, ...) inte längre tillgodosåg alla krav som stora komplexa program ställde med avseende på underhåll, förnyelse och ändringsbarhet. Ingen kunde sätta sig in i, ändra och vidareutveckla ett stort program om programmeraren hade lämnat företaget. Det innebar ett enormt slöseri med resurser. Dessutom utvecklades hårdvaruteknologin så snabbt att program som kunde köras på de allt mer avancerade datorerna blev allt större och mer komplexa, speciellt när det gällde grafiska tillämpningar. Mjukvaruteknologin utvecklades inte alls i samma takt. För att lösa alla dessa problem uppkom den nya programmeringsfilosofin *objektorienterad programming* (OOP) som en vidareutveckling av den traditionella *procedurala programmeringen*.

1983 presenterade Bjarne Stroustrup programmeringsspråket C++. Han behöll hela C och la till de nya objektorienterade elementen, bl.a. klassbegreppet, som hade redan funnits t.ex. i *Simula*, ett norskt programmeringsspråk från 1967 som i sin tur var en direkt utbyggnad av *Algol* (*Algorithmic language*). Simulas klasser hade "glömts bort". Den ovan beskrivna problematiken på 70-talet gjorde att man kom ihåg dem. Förhållandet mellan C och C++ illustrerar bäst den "nya" filosofin tilläggskaraktär:



Dvs C är en *delmängd* av C++. Därför gäller all C-kod även i C++, men inte tvärtom. En C++ kompilator kan kompilera all C-kod, men inte tvärtom. Så den som lär sig C++ lär sig automatiskt C. Delmängdrelationen mellan C och C++ är unik bland högnivåspråken.

C++ är ett kraftfullt och populärt programmeringsspråk, vars styrka ligger i textbaserade konsolapplikationer. Inte att C++ vore principiellt olämpligt för grafiska tillämpningar, bara att det är lite jobbigt att skriva C++ kod för att åstadkomma grafik. En anledning är att, när C++ skapades, hade grafiska tillämpningar bara en begränsad spridning inom IT. Med utvecklingen av webbens grafiska miljö, med spridningen av Windows och grafiska användargränssnitt blev grafiken dominant. Idag har C++ fått en renässans med uppkomsten av *IoT* och *inbyggda system* p.g.a. sin snabbhet och maskinnära egenskap.

Unicodes historia

90-talet **Unicode** är inget programmeringsspråk utan en internationell teckenstandard för utvidgning av ASCII-tabellen. Unicode är av betydelse för programmeringshistorien därför att den är en milstolpe mellan textbaserade språk och grafiska tillämpningar. Det är ingen slump att övergången av det textbaserade operativsystemet DOS till det fönsterbaserade Windows faller i samma period. Det blev nödvändigt att ställa upp en teckenkodningsstandard för grafiska miljöer.

För en utförlig behandling av Unicode hänvisas till avsnitt **2.3 Unicode** (sid 29).