

Programspråket C

Fördelar

C är idag ett av de viktigaste och mest använda programspråken. Detta beror givetvis till stor del på C:s (verkliga och inbillade) förtjänster.

□ *C är effektivt*

C har rykte om sig att ge snabbare (och kompaktare) program än andra högnivåspråk, vilket är delvis sant, delvis en myt, ett måhända kontroversiellt påstående som kräver en lite längre utredning.

C:s upphovsmän hade som målsättning att det skulle vara möjligt att relativt lätt konstruera C-kompilatorer som gav effektiv maskinkod. Därför undvek man sådant som var svårt att översätta till effektiv maskinkod. De avstod även från att införa automatiska feltester (såsom gränskontroller) under exekvering, eftersom sådana skulle kunna dra ned hastigheten (och öka kodstorleken).

Av dessa skäl är det inte förvånande att C-kompilatorer ofta ligger bra till i fråga om den genererade maskinkodens effektivitet vid jämförelser med andra kompilatorer. Det är emellertid även sant att det finns FORTRAN-, Modula-2- och Pascal-kompilatorer som kan ge maskinkod i paritet med de bästa C-kompilatorerna. Maskinkodens kvalitet beror faktiskt betydligt mindre på programspråket än de flesta tror. Skillnaderna bestäms främst av andra faktorer: kompilatorkonstruktörernas skicklighet, den tid som stått till förfogande för att utveckla produkten (nyare versioner brukar ge bättre kod än äldre) och vad som prioriterats (kodkvalitet eller kompileringstid – effektiv maskinkod kräver optimering och därmed längre kompileringstider). Till syvende och sist är det kompilatorns kvalitet och optimeringsmöjligheter som är avgörande, inte programspråket. Detta innebär också att effektiviteten inte bör vara ett huvudargument när man väljer C framför andra programspråk (se även avsnittet om kompilatortester nedan).

□ *C är portabelt*

C är väl standardiserat (i motsats till t ex Pascal, som finns i ett otal dialekter). Ett framgångsrikt internationellt standardiseringsarbete, organiserat av American National Standards Institute (Amerikanska Standardiseringskommissionen, ANSI), har bidragit till detta. ANSI-standarden behandlar fö uttryckligen portabilitetsfrågor, dvs fastställer hur maskin- och kompilatorspecifika detaljer ska hanteras. Idag har de flesta programvaruhus som levererar C-kompilatorer förbundit sig att följa ANSI-C.

Standardiseringen, samt det faktum att det idag finns acceptabla C-kompilatorer till de flesta datorer, innebär att det normalt är relativt enkelt att flytta C-program från en dator till en annan.

C understödjer således portabiliteten (även om det är möjligt att skriva icke portabla C-program). Flyttbarheten är ett av de tyngsta argumenten för C. Förutom Ada (som dock inte är lika spritt) finns knappast något annat programspråk som kan tävla med C i detta avseende.

❑ *C är kraftfullt*

C tillåter rekursion, ger möjligheter att skapa nya datatyper (arrayer, poster osv), har ett stort antal operatörer, medger maskinnära programmering (lågnivåprogrammering) m m.

❑ *C är flexibelt*

C lägger få restriktioner på programmeraren. Mycket som skulle vara tabu i andra programspråk kan utföras i C. Avsaknaden av "inbyggda begränsningar" gör att C kan användas för så gott som alla slags tillämpningar (därmed inte sagt att C alltid är det "lämpligaste" programspråksvalet).

❑ *C:s standardbibliotek*

Med varje C-kompilator ska följa ett standardbibliotek med en rik uppsättning funktioner för in- och utmatning, strängbearbetning, dynamisk minneshantering m m.

❑ *C:s knytning till UNIX*

Eftersom lejonparten av UNIX är skrivet i C, passar de två särskilt bra ihop. Det är t o m så att C-kunskaper krävs för att utnyttja vissa UNIX-verktyg.

Nackdelar

Som alla programspråk har C sina svaga sidor.

❑ *C är inte blockstrukturerat*

C understödjer strukturering av programkod och data, men det är inte *blockstrukturerat*, ty funktioner får inte (som i Pascal, Modula-2 och Ada) definieras inne i andra funktioner, dvs alla funktioner befinner sig på "samma nivå" (funktioner kan inte "inkapslas" i andra funktioner).

❑ *C-program är svåra att få felfria*

C:s flexibilitet, som bl a hänger ihop med den relativt sett svaga typkontrollen, är också C:s akilleshäla. Många felaktigheter som t ex en Pascal-kompilator skulle fånga upp, släpper C-kompilatorer igenom utan protester, även om den nya ANSI-standarderna har stramat åt C-språket i dessa avseenden.

C saknar inbyggda mekanismer, såsom de i Modula-2, för typ- och versionskontroll mellan separatkompileerade moduler, vilket gör C-program mycket sårbara för integrationsfel.

C innehåller många fallgropar. En del av dessa är t o m implementeringsberoende, t ex har C-kompilatorer frihet att ändra beräkningsordningen av vissa uttryck (detta för att kunna optimera beräkningarna). Ämnet är så omfattande att det författats speciella böcker om dessa problem.

❑ *C:s konstruktioner kan vara svåra att förstå*

Även om C ur flera aspekter är ett litet språk (det har t ex bara c:a 30 nyckelord), har det ett antal språkelement som saknas i andra vanliga programspråk och som därför lätt missförstås.

❑ *C-program kan vara svåra att modifiera*

C skapades vid en tidpunkt då interaktiv programmering av det slag vi idag vant oss vid var okänd. För att minimera den tid som behövdes för att skriva in och redigera program, hölls C medvetet så koncist och kortfattat som möjligt. Detta kan dock göra C-program svårlästa och därmed svåra att ändra. En urskillningslös användning av C:s kraftfulla konstruktioner kan göra programmen nästan omöjliga att begripa. Det förefaller också som om det bland vissa C-programmerare blivit närmast en sport att åstadkomma så kryptiska program som möjligt. Detta otyg underblåses tyvärr ibland av aningslösa redaktörer på datatidskrifter, som verkar bli mer "imponerade" av en programlistning ju mindre de själva begriper av densamma – skärpning om vi får be!

Slutsatser

C är definitivt inget nybörjarspråk. Det är ett språk för programmerare. Den grundläggande filosofin bakom C är att programmeraren vet vad hon/han gör. C ger stora friheter, men lägger också ett tungt ansvar på programmeraren, som inte kan förlita sig på att kompilatorn fångar upp de flesta felen. Konsekvent användning av funktionsprototyper och övriga förbättrade möjligheter till skärpt typkontroll m m, som den nya ANSI-standarderna erbjuder, liksom utnyttjande av bra programmeringsverktyg (såsom avlusningsprogram och *lint*), kan reducera felriskerna, men det är också möjligt att med slarv eller obetänksamhet göra mycket grova fel. Att skriva korrekta C-program kräver självdisciplin och goda programmeringsvanor. För den som förmår uppfylla dessa krav kan C vara ett mycket effektivt och uttrycksfullt programspråk, som passar en mängd olika tillämpningar.