PROGRAMMERING
VISUAL C++
2008

DOCENDO

Innehållsförteckning

1 Allmänt om C++

I detta kapitel

I detta kapitel kommer du att få veta lite om C++ och dess historia. Du kommer också att få reda på lite om vad som kommer att krävas av dig för att bli en fullfjädrad programmerare. Dessutom blir det lite grunder i terminologin och information om hur programmering fungerar.

Historia

Programmeringsspråket C utvecklades 1969–1973 av Bell Laboratories i samband med utvecklingen av operativsystemet Unix. Föregångaren till C var ett programspråk vid namn B.

På 80-talet förfinade Bell Laboratories språket och gjorde det objektorienterat. Den nya versionen fick det fantasifulla tillägget ”++”. Objektorienterat innebär att man som programmerare tänker i termer av vilka delar, objekt, som finns med i systemet man ska bygga. Icke objektorienterad programmering tenderar att fokusera mer på hur ett problem ska lösas. Därför är det lättare att ”tänka objektorienterat” och därmed skapa program med färre fel i. Det finns också andra fördelar med objektorientering, till exempel att det är lättare att återanvända objekt nästa gång man gör ett liknande system.

Om man förstår C++ så förstår man även C. C och C++ tillhör fortfarande de absolut mest använda programspråken.

Hur fungerar det att programmera?

Hårdvara och mjukvara

Hårdvara och mjukvara talas det ofta om i databranschen, även kallade maskinvara och programvara. Vad är då detta? Helt enkelt kan man förklara det så att hårdvara är ”hårda saker”, det vill säga saker man kan ta på. Mjukvara är själva programmet, till exempel Windows och Word. Det går inte riktigt att ta på Windows eller Word, utan de liknar mer idéer eller funktioner för användaren. För programmeraren, å andra sidan, är ett program en uppsättning instruktioner som talar om för datorn hur den ska bete sig.

Ett program kan också kallas en applikation; det är samma sak.

Fel i mjukvaran – avlusning

Ibland när programmeraren gör ett fel så blir det märkbart för användaren i slutändan. Då talar man om en ”bugg”. En bugg kan leda till allt ifrån irriterande brister, till exempel att det plötsligt inte går att zooma i Word, till mer allvarliga fel som att hela Windows bryter ihop och visar en blå skärm med felinformation. När man letar efter ett fel med avsikten att ta bort det så kallas det för ”debugging” eller avlusning.

Bugg kommer av engelskans ”bug”, och det är som bekant engelska för insekt. Som kuriosa kan man nämna en lustig anekdot om hur det ordet kommit att användas för programmeringsproblem, och det har att göra med att de första datorerna såg helt annorlunda ut än dagens. Då användes datorer som fyllde flera stora rum och som drog enorma mängder ström med sladdar och radiorör om vartannat. En dator då hade de ungefärligen samma kapacitet som en vanlig, enkel digital klocka har idag. Ett vanligt fel som uppstod vid driften av dessa datorer var att kackerlackor och andra insekter nästlade sig in mellan strömförande ledningar. Samtidigt som de gjorde sig själva en otjänst och blev elektrifierade så lyckades de kortsluta en del av datorn, och programfel tillsköt. Därav bugg.

CPU

En mycket central del i varje dators hårdvara är dess **CPU**. CPU betyder *Central Processing Unit*, eller centralprocessor. Det är CPU:n som steg för steg utför alla instruktioner som programmeraren matat in för hand.

Centralprocessorn innehåller logiska enheter för att kunna utföra addition, subtraktion, multiplikation med mera. Det finns också funktioner för datakopiering och jämförelser, till exempel större än, mindre än eller lika med, lika med, skilt från, et cetera.

En CPU kan bara läsa och utföra ett särskilt sorts kodsystem. Detta kodsystem kallas maskinkod och varierar mellan olika märken och olika sorters CPU:er. Maskinkod är helt obegripligt för den oinvigde: till exempel betyder `1000101111000011` ”flytta data från b till a” för en Pentium-kompatibel CPU; fullkomligt meningslöst och obegripligt alltså. Det viktiga du behöver veta här är bara att maskinkod är en lista med instruktioner som CPU:n kan förstå.

Källkod

För att slippa skriva ettor och nollor så skriver programmeraren ”källkod”, eller ”kod”, som sedan översätts till maskinkod. Källkoden är text i en helt vanlig textfil, men språket man använder har en avsevärt enklare grammatik än vad vi är vana vid. Även i dess mest avancerade form är den väldigt basal och mycket långt ifrån vanlig enkel svenska eller engelska.

Källkoden skriver man in ungefär som vanlig text i en ordbehandlare: rad för rad, i små stycken eller satser. Ett komplett program kan innehålla allt ifrån 10 rader till miljontals rader. En programsats motsvarar ungefär ”en mening” i vanlig skrift. En sats är helt enkelt en grupp med ord som kompilatorn kan förstå.

Nyckelord

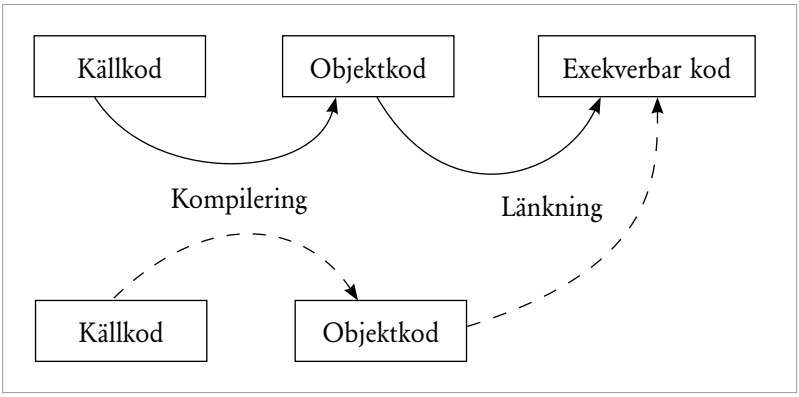
Specifika ord som används i ett språk kallas ”nyckelord”. Dessa ord är i C++ på engelska och är ofta koncisa. Några exempel: **`if`**, **`while`**, **`switch`** och **`case`**. I andra programmeringsspråk används andra nyckelord.

Kompilering

För att kunna köra programmet så måste man ”översätta” källkoden till maskinkod som CPU:n kan förstå. Den här översättningen från källkod till maskinkod kallas *kompilering*. Programmet som kompilerar heter kompilator.

Ofta används nuförtiden ett och samma verktyg för att skriva in källkod, kompilera och avlusa. Förr användes ofta ett vanligt textredigeringsverktyg för källkoden och en kompilator som man startade från kommandotolken; om man vill vara lite retro så kan man fortfarande göra på detta vis. Den här kursen kommer att gå in mer djuplodande på ett integrerat verktyg av Microsoft: Visual C++ 2008 Express Edition.

Kompileringen sker egentligen i två steg: kompilering och länkning. Det beror på att man oftast har flera källkodsfiler som man vill slå ihop till ett enda program. Till exempel kanske man har en källkodsfil för att rita ut 3D-objekt och en annan för TCP/IP-kommunikation.



Verktyg

Det finns massor av olika verktyg för att jobba integrerat med källkod, kompilering, avlusning, et cetera. Dessa verktyg brukar benämnas *utvecklingsverktyg* eller *utvecklingsmiljöer*, på engelska **IDE** = *Integrated Development Environment*. Här är några få av de vanligaste idag:

- Microsoft Visual C++
- Code::Blocks
- Eclipse
- NetBeans
- GNU Emacs (textredigerare) + GNU Compiler Collection (kompilator)

Programmeringsspråk

Det finns många olika programmeringsspråk; vissa riktar sig mer mot hårdvaran, så kallade lågnivåspråk, andra siktar på att vara enklare att producera och förstå, så kallade högnivåspråk. Varje språk har sin egen kompilator, dock finns det utvecklingsmiljöer som stödjer användning av flera programspråk samtidigt. Microsoft Visual Studio 2008 är ett sådant.

Ett lågnivåspråk är ofta plattformsberoende, det vill säga de fungerar bara för en särskild sorts maskinkod och därmed bara en särskild sorts CPU. Lågnivåspråk kan i sällsynta fall vara mycket kraftfulla när man behöver specialanpassa en liten programsnutt till en specifik plattform. Oftast är dock så ej fallet. Programkod i ett lågnivåspråk är dessutom mer specialiserad och därigenom svårare att producera och förstå.

Programmeringsspråk som riktar sig mot många plattformar och har som mål att vara enklare att producera och förstå kallas för högnivåspråk. C++ är i första hand ett högnivåspråk, men kompromisser gör att det även kan leverera högsta prestanda.

Programexekvering

När ett program exekveras (körs) så kan man tänka på det som att CPU:n läser och utför koden uppifrån och ner, från vänster till höger. Det hela fungerar lite som att läsa en instruktionsbok (till exempel med instruktioner för montering av en möbel).

Loopar

Vissa instruktioner kan få CPU:n att ”hoppa” i koden, det vill säga fortsätta exekveringen på en annan position. Detta kan till exempel utnyttjas till att upprepa ett stycke av koden flera gånger fast med olika indata. Då är det vanliga att exekveringen börjar om ifrån början på satsen. Det kallar programmeraren för en loop eller en slinga. Ett vardagsexempel på en loop kan man ofta finna i visor. Verserna i en visa upprepar man inte men på refrängen händer det att man lägger en loop: helt enkelt läser och utför stycket flera gånger.

Ett loop-exempel till kan vara att leta rätt på ett namn i en lista. Proceduren för att leta rätt på ett namn är detsamma för varje rad i listan med namnen; det är bara data (namnen) som varierar från en rad till annan.

Oftast kan man urskilja tre övergripande delar i ett mer avancerat program: startfasen, den interaktiva fasen och avslutningsfasen. Under den interaktiva fasen loopar ofta applikationen i väntan på indata från tangentbordet och musen.

Ännu ett exempel på varför loopar behövs: se hur Windows fungerar! Om programkoden för Windows exekverades rakt igenom, allt på en gång, så skulle det ju avslutas direkt efter att ha startats.
Nu fungerar det inte så eftersom Windows har en interaktiv loop som väntar på indata från användaren.

Inspiration

Det finns inga programmeringsuppgifter man inte kan lösa med C++. Programmeringsspråket uppmuntrar inte alltid till det smartaste sättet att lösa en uppgift, så det kräver lite mer av programmeraren i vissa fall. Men resultatet är ofta rakt på sak och prestandan på programmen blir mycket god.

De allra flesta prestandakrävande program som görs idag använder C++.
Det gäller allt ifrån dataspel till avancerade beräkningar för forskningssyfte.
Dataspel kan till synes vara enkel underhållning, men faktum är att de är något av det svåraste man kan ägna sig åt i databranschen – men vansinnigt kul!

Målet med din programmeringskurs måste vara att det ska bli roligt. Det är då man lär sig fortast, och det är då man blir riktigt, riktigt bra på det. Ett av delmålen med detta material är därför att göra det så roligt som det bara går. Försök själv att göra uppgifter och program i materialet så underhållande som möjligt för dig själv. På så sätt kommer du att få ut mesta möjliga av det.

I slutet av kursen kommer du att göra en fraktalzoomare, det vill säga en applikation som ritar ut en synnerligen vacker och kaosartad matematisk funktion. Du kommer att få hjälp med hur du går vidare efter kursen och det medföljer fler roliga övningsexempel (public domain) som du kan använda och modifiera som du finner bäst.

Styrkan i C++

Genomslagskraften i C++ beror på många faktorer, men det är framförallt dessa:

- Det är objektorienterat.
- Mycket hög prestanda.
- God kontroll. Man kan delvis tänka sig C++ som ett lågnivåspråk på grund av den strikta kontrollen över både vad programkoden gör och hur datahållningen ser ut.
- Mycket stor basfunktionalitet. Man behöver inte återfinna hjulet.
- Det är enkelt att koppla ihop programkod i C++ med så kallade bibliotek skrivna i andra programspråk.
- Det är standardiserat och fungerar likadant på alla datorer.

Övningsuppgifter

Övning 1.1 På 80-talet utvecklades C till C++. Vad var den största nyheten?

Övning 1.2 Vad är hårdvara och vad är mjukvara? Ge även exempel på var och en av dem.

Övning 1.3 Vad kallas ett programmeringsfel?

Övning 1.4 Vad heter den centrala exekverande enheten i datorn?

Övning 1.5 Vad kallas en lista med instruktioner som den centrala exekverande enheten i datorn kan förstå?

Övning 1.6 Vad kallas processen som omvandlar källkod i C++ till instruktioner som centralprocessorn kan förstå?

Övning 1.7 Man skiljer på två olika nivåer på programspråk. Den ena är specialanpassad till en viss sorts CPU, medan den andra är anpassad för att funka med många plattformar och vara enklare att skriva och läsa. Vad kallas de två nivåerna?

Övning 1.8 Körs alltid programkoden sekventiellt, eller kan du som programmerare avgöra vilken del som körs härnäst?

2 Utvecklingsverktyget

I detta kapitel

Här kommer grunderna för hur du arbetar med programmering i allmänhet och med utvecklingsverktyg för C++ i synnerhet. Vi har valt utvecklingsverktyget Microsoft Visual C++ 2008 Express Edition (Visual). Om du arbetar i ett annat program kan du ändå känna igen dig och göra övningarna. Bilder, exempel och instruktioner i detta kapitel visar Visual. Men alla övningar i boken är generella och kan göras med en annan version, och även med ett helt annat verktyg.

Utvecklingsverktyg

Visual (eller något annat utvecklingsverktyg efter eget val) använder du för att skriva källkod i, kompilera den, köra programmen du skrivit och sist, men inte minst, för att avlusa programmen. Du får också veta lite om huvudfunktioner och du kommer att få lära dig att skapa en enkel konsolapplikation och hur du öppnar ett befintligt projekt.

Säkerhetskopiera

Börja med att kopiera övningsfilerna som medföljer materialet. Det bästa är om du hela tiden arbetar med lokala kopior. Lägg till exempel källkoden någonstans under **Mina dokument**; exempelvis i den mapp som skapas automatiskt av Visual vid installationen, mappen **Mina dokument\Visual Studio 2008**. Om det är lämpligt för dig så skapar du också en ny mapp kallad: **C++ A** till vilken du kopierar övningsfilerna.

Arbetssätt

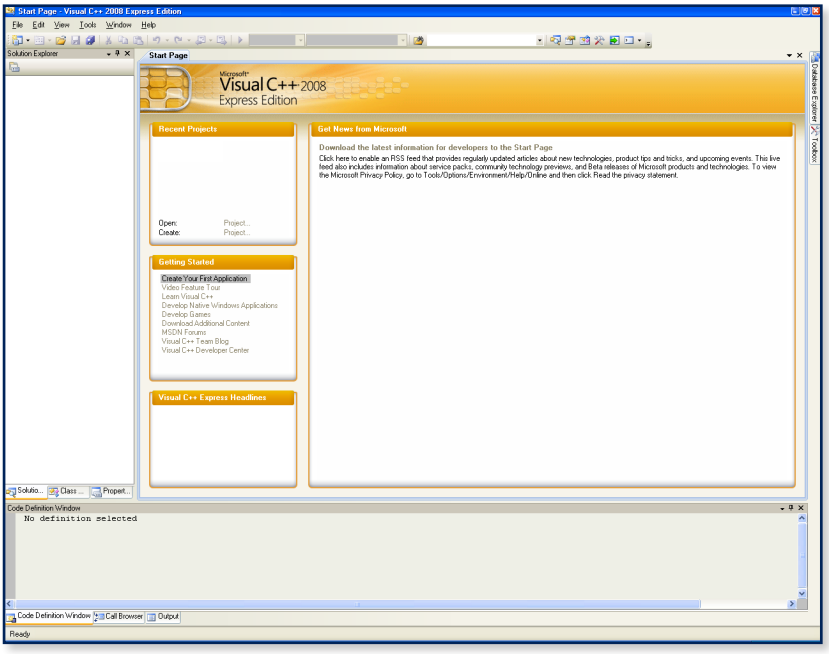
När du programmerar så kommer du i mycket stor utsträckning att använda dig av tangentbordet. Det beror på att arbetsgången ser ut som den gör. När en programmerare jobbar brukar det se ut ungefär så här: man skriver korta kodsnuttar, kompilerar, rättar några fel, kompilerar, funderar ett par sekunder, lägger till ytterligare några rader och kompilerar igen. Det hade helt enkelt blivit för irriterande och långdraget om man var tvungen att använda musen i alla moment. Därför kommer du snabbt att lära dig de vanligaste kortkommandona.

Starta programmet

Du ska börja med att starta programmet (utvecklingsverktyget).

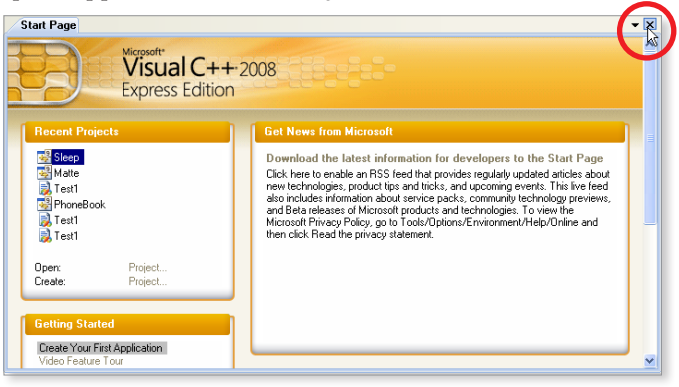
1. Klicka på **Start**-knappen, välj **Alla Program**, **Microsoft Visual C++ 2008 Express Edition**, välj **Microsoft Visual C++ 2008 Express Edition**.

När programmet startas för allra första gången kan det se annorlunda ut mot hur det ser ut då du startar det fortsättningsvis.



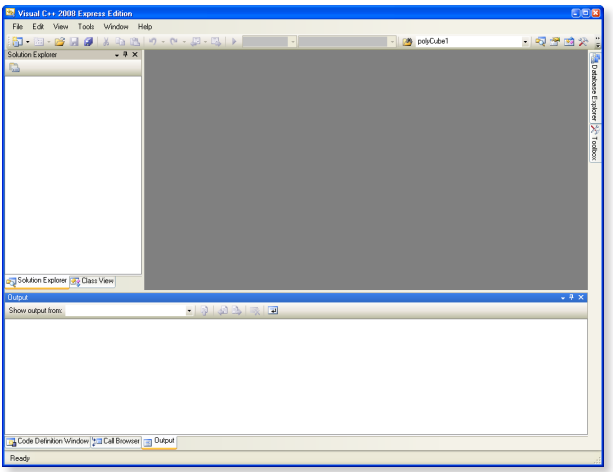
Ser programmet annorlunda ut kan du hoppa över nästföljande steg och gå direkt till avsnittet Arbetsytan.

2. Klicka på knappen **Close** i övre högra hörnet av startsidan.



Arbetsytan

Du ser nu en tom arbetsyta:



Benämningar

De vanligaste delarna du kommer att använda dig av är:

- Textytan
- Arbetsfältet
- Utdatafältet
- Verktygsfältet