1 Inledning

För att man skall kunna köra (exekvera) ett program på en dator krävs det att man har en fil skriven på ett sådant sätt att datorn förstår vad som avses, dvs. man behöver en fil som endast använder sig av instruktioner som finns tillgängliga på aktuell dator och aktuellt operativsystem. För att själv slippa att skriva ettor och nollor har man olika hjälpprogram som omvandlar sin egna kod (t.ex. C-kod) till assemblerkod och maskinkod. Denna omvandlingsprocess sker med hjälp av en assemblator, en kompilator och en länkare. I denna laboration skall fås en första inblick i hur man skriver och editerar ett program, hur man skapar ett projekt i Visual C++, samt hur man kompilerar och länkar sitt program.

Vidare skall ni få en inblick i debugging av program.

2 Programutveckling i Visual C++

2.1 Att komma igång - Skapa ett nytt projekt

1) Skapa en katalog på H-disken för laborationen, t.ex. H:\RTP\LAB1\.
2) Starta Visual C++ 6.0 från Windows startmeny.
3) Klicka på File – New.
4) Markera ”Win32 Console Application” under fliken ”Projects”. (Detta innebär att man skall utveckla ett program som endast skriver till och från ett console fönster, dvs. att man inte skall utveckla något grafiskt program för Windows.
5) Se till att det står rätt sökväg, dvs. den katalog som ni just skapat (t.ex. H:\RTP\LAB1\), under ”Location”. Om det inte redan gör det så kan man navigera sig genom att trycka på knappen med tre punkterna, belägen till höger om ”Location”.
6) Under ”Project Name” skriver ni t.ex. Upp1 och trycker på ”OK”.
7) På frågan vilken typ av Console Application ni vill skapa svarar ni ett tomt projekt.
8) Nu bör det ha skapats en ny katalog med namnet Upp1 i katalogen H:\RTP\LAB1\, kolla detta.
9) Projektet är numera uppsatt, och det är dags att börja programmera. För att programmera behöver man skapa en källkodsfil (.c-fil) i vilken man skriver sin kod. Detta görs på liknande sätt som när man skapar ett nytt projekt.
10) Tryck på File – New.
11) Markera ”C++ Source File” under fliken ”Files”.
12) Döp filen med ett lämpligt namn, t.ex. lab1_u1.c, under ”File Name”. OBS! Filerna skall döpas enligt ”Namn”.c. Det är viktigt att man skriver ”.c” för annars skapas en fil för C++, vilket kan leda till problem för tillfället. Tryck på ”OK”.
13) Filen bör numera finnas på den stora skrivytan under Visual C++.
14) Skriv in följande testprogram:

```
#include <stdio.h>

int main(void)
{
   printf(”Hello World!\n”)

   return(0);
}
```

15) Spara din fil genom File – Save.
16) Kompilera din fil genom att trycka på Build – Compile (lab1_u1.c), vad händer?
17) Om du skrev in texten precis som det står i laborationshandledningen bör kompilatorn svara att koden innehåller syntax fel. För att snabbt se var kompilatorn hittat felet kan man dubbelklicka på felmeddelandet i det undre fönstret.
18) Åtgärda felet och kompilera igen. Om felet är korrekt åtgärdat bör kompilatorn svara: ”upp1_1.obj - 0 error(s) and 0 warning(s)”.
19) För att få en körbar fil återstår endast att länka programmet. Länkningen sker genom att trycka på Build – Build Lab1.exe. TIPS: Man behöver inte köra kompileringskommandot utan det räcker att köra länkningskommandot, då programmet kompilerar alla icke kompilerade filer före länkning.
20) Kör programmet genom att trycka på Build – Execute Lab1.exe, vad händer?

21) Lägg till följande två rader efter raden printf(”Hello World!\n);:

```
printf(”My name is Ola”);
printf(”and I live in Halmstad.”);
```

22) Kompilera och länka, vad händer?
23) Tanken som programmeraren hade när han lade till de två raderna var att få utskriften från programmet enligt följande:

```
Hello World!
My name is Ola
and I live in Halmstad.
```

Hur kommer det sig att utskriften inte blir som det var tänkt? Korrigera koden så att utskriften blir enligt önskemål och kör programmet igen.

24) Skriv om programmet så att önskat resultat skrivs ut men där du endast använder en printf(…)-sats.
25) OBS: Glöm inte att spara ditt ”Workspace”, dvs. den arbetsyta som du för tillfället arbetar med. Detta görs för att lättare komma igång nästa gång man vill fortsätta på sitt arbete då det räcker att öppna den sparade arbetsytan för att fortsätta att på projektet. Om man trycker på File – Save Workspace sparas arbetsytan i katalogen H:\RTP\LAB1\Upp1\ och får namnet Upp1.dsw, kontrollera detta.

3 Debugging

3.1 Användande av Debugger

Att effektivt lokalisera fel i olika program är troligtvis det mest viktiga en programmerare kan lära sig. Beroende på vad man programmerar och i vilket språk man programmerar så står olika bra hjälpmedel till buds. I Visual C++ Studio finns en mycket bra Debugger integrerad i systemet, med vilken man kan exekvera sitt program en rad i taget och samtidigt kontrollera vilka värden olika variabler har. Man kan också kontrollera huruvida programmet hoppar till rätt instruktion, dvs. om programmet uppför sig enligt önskemål, eller inte.

Man kan generellt säga att kompilatorn hittar fel i syntaxen och med Debuggerns hjälp kan man hitta logiska fel, så ta för vana att kontrollera dina program i en Debugger.

Uppgifterna 3.3 – 3.5 går ut på att lära sig att använda Debuggern samt lokalisera och åtgärda uppkomna logiska fel i några olika program. Observera att samtliga program går igenom kompilering och länkning utan att felen upptäcks.

Utgå från filen ”lab3_33.c” som finns att ladda hem från kursens hemsida. Kompilera och länka som ett nytt projektet. Tanken med programmet är att det skall räkna upp en variabel från noll till nio, kontinuerligt skriva ut värdet av variabeln för att sedan räkna ned variabeln igen och kontinuerligt skriva ut värdet, dvs. *önskad* utskriften från programmet skall se ut enligt följande:

```
Påbörjar uppräkning
Demoprogram för utskrift - Uppåt – 0
Demoprogram för utskrift - Uppåt – 1
Demoprogram för utskrift - Uppåt – 2
Demoprogram för utskrift - Uppåt – 3
Demoprogram för utskrift - Uppåt – 4
Demoprogram för utskrift - Uppåt – 5
Demoprogram för utskrift - Uppåt – 6
Demoprogram för utskrift - Uppåt – 7
Demoprogram för utskrift - Uppåt – 8
Demoprogram för utskrift - Uppåt – 9

Påbörjar nedräkningen
Demoprogram för utskrift - Nedåt – 9
Demoprogram för utskrift - Nedåt – 8
Demoprogram för utskrift - Nedåt – 7
Demoprogram för utskrift - Nedåt – 6
Demoprogram för utskrift - Nedåt – 5
Demoprogram för utskrift - Nedåt – 4
```

```
Demoprogram för utskrift - Nedåt – 3
Demoprogram för utskrift - Nedåt – 2
Demoprogram för utskrift - Nedåt – 1
Demoprogram för utskrift - Nedåt – 0

Programmet är numera slut
```

Kör programmet och se vad som händer innan du går vidare!

Brytpunkter: När man kör programmet ser man att utskriften inte överensstämmer med den önskade, hur kan detta komma sig? Svaret är naturligtvis att programmet innehåller logiska fel, dvs. programmeraren har skrivit sitt program med felaktig logik. När man kör programmet ser man också att en del av utskriften är korrekt, dvs. den första delen av utskriften, vilket får en att misstänka att felet ligger i den andra loopen, dvs. while-loopen. För säkerhets skull skall vi även debugga den första delen av programmet.

Man kan (nästan) var som helst i programmet sätta *brytpunkter*, vilka gör så att programexekveringen stannar och man kan kontrollera att värdena på aktuella variabler har önskade värden. Sätt en *brytpunkt* på raden ” printf("Påbörjar uppräkning\n");” genom att klicka på höger musknapp på raden ” printf("Påbörjar uppräkning\n");” – Välj därefter ”Insert/Remove Breakpoint” varefter *brytpunkten* läge markeras med en röd cirkel i vänstermarginalen.

Kör programmet fram till en brytpunkt: Starta debuggern genom att trycka på ”Build” -> ”Start Debug” -> ”Go” (Man kan också trycka på F5), varefter programmet startar och kör fram till den första *brytpunkten*. Var programmet för tillfället är markeras med en gul pil i vänstermarginalen i koden.

Kontrollera variabler: Det finns tre sätt att kontrollera värdet av en variabel, nämligen; 1) Hålla muspekaren över den variabel man vill kontrollera varpå värdet av variabeln visas i anknytning till muspekaren – testa genom att hålla muspekaren över variabeln ”i” -, 2) I nedre vänstra hörnet visas kontinuerligt värdet av det senast uträknade värdet eller variabeln – kontrollera att värdet på ”i” visas där - och 3) Markera variabeln ”i” med hjälp av musen och dra och släpp den i det nedre högra hörnet, varefter variabeln stannar där till dess att man själv tar bort den. Viktigt är också att en variabel man lagt till i det nedre högra hörnet uppdateras allteftersom man använder den i programmet – Testa genom att markera, dra och släpp variabeln ”i” i det nedre högra hörnet.

Stega sig genom programmet: Genom att trycka på ”Debug” -> ”Step Over” (Man kan också trycka på F10) exekverar man nästa rad i programmet – detta kallar man att steppa sig igenom programmet. Steppa så att markören hamnar på raden ”printf("Demoprogram för utskrift - Uppåt - %d\n", i);”, kontrollera värdet på ”i”, förvissa dig om att det är rätt och steppa vidare. Steppa vidare (håll hela tiden koll på vad värdet av ”i” är, så att du vet att rätt sak skriv ut) så att du hamnar på raden ”printf("\nPåbörjar nedräkningen\n");”. Kontrollera värdet av ”i”! Steppa fram till raden ”printf("Demoprogram för utskrift - Nedåt - %d\n", i--);” och om man kontrollerar ”i” så ser man att ”i” har värdet 10, vilket således kommer att skrivas ut på skärmen, vilket också är fel! Nu när man lokaliserat felet är det bara att rätta till programmet antingen genom att ändra raderna;

```
”while(i){”          =>         ” while(i--){” och
”printf("Demoprogram   för   utskrift   -   Nedåt   -   %d\n",   i--);”   =>
”printf("Demoprogram för utskrift - Nedåt - %d\n", i);”
```

eller genom att dekrementera ”i” före utskriften, dvs. följande ändring;

```
”printf("Demoprogram   för   utskrift   -   Nedåt   -   %d\n",   i--);”   =>
”printf("Demoprogram för utskrift - Nedåt - %d\n", --i);”.
```

Utför ändringarna och kör programmet igen!

3.2 Felsökning av Program

Utgå från filen ”lab3_34.c” som finns att ladda hem från kursens hemsida. Tanken med programmet är att man skall kunna mata in två flyttal, fNum1 och fNum2, varefter programmet beräknar kvoten fNum1 / fNum2 och presenterar den som ett flyttal på skärmen. Inmatningen, och beräkningarna, skall pågå till dess att användaren matar in värdet 0.0 på variabeln fNum2.

Lokalisera och åtgärda felen med hjälp av debugger, så att programmet får önskad funktionalitet.

3.3 Felsökning av Program

Utgå från filen ”lab3_35.c” som finns att ladda hem från kursens hemsida. Tanken är att programmet skall läsa in ett tecken i taget till dess att användaren avslutar genom att trycka på bokstaven ’q’ eller ’Q’. Om någon av bokstäverna ’a’, ’b’, ’s’ eller ’e’ matas in skall programmet svara med motsvarande versal, dvs. om ’a’ matas in svarar programmet med ’A’ och inget annat. För övriga tecken, förutom ’q’, ’Q’, ’a’, ’b’, ’s’ eller ’e’ skall programmet svara att tecknet inte tas om hand om.

Avgör om programmet innehåller några fel genom några vettiga tester och om det finns några fel – åtgärda dem!

4 Sammanfattning

Efter laborationen skall studenten kunna; 1) skapa ett nytt projekt, 2) kompilera och länka ett projekt, 3) skapa ett enkelt program. dvs. veta att programexekveringen startar i funktionen ”main(..)”.

Ni skall också kunna hantera och lokalisera fel med hjälp av en debugger. Glöm inte att debuggern är programmerarens bästa vän!

Den här laborationen är ett hopkok av andra laborationer utvecklade av Ola Bengtsson (Ola.Bengtsson@ide.hh.se, http://www.hh.se/staff/boola). Tack till Ola.