Introduktion till C-programmering, VT2004

18 januari 2004

1 Lärare

Namn	E-post	Rum	Postfack
Mikael Pettersson	`mikpe@csd.uu.se`	PB 1318	PB, hus 1, våning 4, nr 86

2 Översikt

Kursen avser att förmedla grundläggande färdighet i imperativ programmering med programspråket C.

3 Kursinnehåll

Maskinmodell: processor, register, minne, programflöde. Imperativa programspråk: relation till maskinmodellen, skillnad mot funktionella programspråk. Uttryck: variabler, konstanter, operatorer. Styrande satser. Deklarationer av variabler och funktioner. Programstruktur. Standardbibliotek, textuell I/O, strängar. Kommandoradsparametrar. Bastyper och sammansatta typer (arrayer och poster). Pekare. Dynamiska datastrukturer. Unionstyper. Fallanalys.

4 Kurslitteratur

Valfri lärobok i C-programmering. För seriöst bruk rekommenderas Harbison & Steele, "C: A Reference Manual" eller Kernighan & Ritchie, "The C Programming Language", men de är inte läroböcker i allmän programmering.

På `http://www.csd.uu.se/~mikpe/teaching/cintro/2004/` finns denna kursinformation samt kopior av föreläsningsanteckningar.

5 Examination

Kursen examineras genom tre laborativa inlämningsuppgifter, i vilka enklare programmeringsproblem skall lösas i C. Inlämningsuppgifterna skall lösas i grupper om två, och lösningarna skall lämnas in skriftligt i kursledarens postfack senast vid angivna deadlines.

Den skriftliga redovisningen skall innehålla en programlistning och ett försättsblad med kursnamn, numret på inlämningsuppgiften, samt studenternas namn och personnummer (utan personnummer kan resultat inte registreras!).

6 Undervisning

Undervisningen ges som föreläsningar, laborationer och handledning. All undervisning är frivillig (utom uppropet).

Kursledaren har handledningstid Fredagar kl 14.00-15.00 under läsperiod 3, men det går också att få handledning via e-post. Observera att handledning inte

är till för att avlusa er kod, utan för att hjälpa er med att lösa specifika problem med t ex verktygen och problemstrukturering.

7 Kursplan

Notera att denna kursplan skiljer sig från ITPs/DVPs preliminära scheman.

Vecka 4		
Mån	Fö1	13–15, Å80101. Maskinmodell, imperativ programmering, variabler, tilldelning, kontrollflöde, procedurer. Exempel.
Vecka 5		
Ons	Lab	15–17, P1515, för ITP.
Tor	Lab	10–12, P1412/P1413, för ITP.
Vecka 6		
Tor	Fö2	13–15, Å80101. Grunderna i C: variabeldeklarationer, uttryck, satser, funktioner, `main()`, enkel input/output. Exempel.
Fre	Lab	13–15, P2510, för DVP.
Vecka 7		
Mån	Fö3	13–15, Å80101. Typer, automatiska konverteringar, array-parametrar till funktioner, pekararitmetik, variablers synlighet och livslängd, skrivskyddade variabler.
Tor	Lab	10–12, P1412/P1413, för ITP.
	Fö4	13–15, P1211, för DVP. Poster, dynamiska datastrukturer. Standardbibliotek, `<stdio.h>` för input/output.
	Lab	13–15, P1412/P1413, för ITP.
Fre		15.00, deadline för inlämningsuppgift 1.
Vecka 8		
Mån	Fö4	13–15, Å80101, för ITP.
Ons	Lab	10–12, P2510, för DVP.
Tor	Fö5	8–10, Å80101. "Avancerad" C: unionstyper, `switch`-satsen, typkonvertering, funktionspekare, preprocessorn, bitfält, icke-lokala hopp (`longjmp`), funktioner med variabelt antal parametrar, klassisk C vs ANSI-C vs ISO-C.
Fre	Lab	13–15, P2510, för DVP.
		15.00, deadline för inlämningsuppgift 2.
Vecka 9		
Ons	Lab	10–12, P1515, för ITP.
	Lab	13–15, P1515. för ITP.
Fre		15.00, deadline för inlämningsuppgift 3.

8 Inlämningsuppgift 1

8.1 Deluppgift A

Implementera en funktion `int gcd(int u, int v)` som beräknar största gemensamma nämnare för två icke-negativa heltal. Utgå från nedanstående definition och konvertera den till en imperativ algoritm, utan användning av rekursion.

$$\gcd(u,\ v) \ = \ \begin{cases} u, & \text{if } v = 0 \\ \gcd(v,\ u \bmod v), & \text{if } v \neq 0 \end{cases}$$

Lägg därefter till en huvudfunktion `main()`, som skall anropa `gcd()` med två godtyckliga värden och skriva ut resultatet mha `printf()`.

8.2 Deluppgift B

Implementera en funktion `int sumvec(int V[], int n)` som beräknar summan av en heltalsvektor `V` med `n` element. Implementationen skall vara imperativ, utan användning av rekursion.

Lägg därefter till en huvudfunktion `main()`, som skall deklarera en lokal vektorvariabel med lämplig storlek, initialisera dess element, anropa `sumvec` med denna vektor och dess storlek som parameterar, och skriva ut resultatet mha `printf()`.

8.3 Deluppgift C

Implementera ett program som tar tre parametrar på kommandoraden. De första två parametrarna är flyttal, den tredje är en sträng `add` eller `mul`. Beroende på utseendet på den tredje strängen skall de två talen antingen adderas eller multipliceras. Resultatet skall skrivas ut till `stdout`. Exempel:

```
uppgift1c 1.5 2.0 mul
```

skall skriva ut `3.0`.

För att lösa uppgiften skall ni använda funktionen `strtod` från `<stdlib.h>` och funktionen `strcmp` från `<string.h>`. Använd `man`-kommandot för att hitta dokumentationen för dessa funktioner, eller titta i er lärobok.

9 Inlämningsuppgift 2

Skriv ett program som sorterar heltal. Indata är en textfil innehållandes ett antal heltal, ett per rad. Utdata är en textfil med samma mängd heltal, ett per rad, i stigande ordning. Dubletter skall tas bort. Indata skall läsas från `stdin`. Utdata skall skrivas till `stdout`.

För att sortera talen skall insättningssortering i en länkad lista användas. Listans storlek skall anpassas under körningen, dynamisk minnesallokering krävs således. Innan programmet avslutas skall allt dynamiskt allokerat minne frigöras.

Den som så önskar får använda ett binärt sökträd istället för en lista.

10 Inlämningsuppgift 3

En enkel interpretator för aritmetiska uttryck skall implementeras. De uttryck som skall hanteras är: heltalskonstanter, negation, addition och multiplikation. Ex: $1 + 2 * -3$.

Steg 1: Definiera en datatyp för representation av dessa uttryck i trädliknande form. Implementera hjälpfunktioner `mk_int_expr(i)`, `mk_neg_expr(e)`, `mk_add_expr(e1,e2)` och `mk_mul_expr(e1,e2)` som skapar träd för de givna typerna av uttryck.

Steg 2: Implementera en funktion `int eval(e)` som tar ett uttryckskträd `e` och beräknar dess värde.

Steg 3: Implementera ett huvudprogram `main()` som först skapar ett uttryckskträd genom anrop till funktionerna från Steg 1, evaluerar det med `eval()`, och till sist skriver ut dess värde. Ni får själva välja vilket uttryck trädet skall representera, men det måste innehålla samtliga former av uttyck som skall hanteras.