Programmering 1

Laboration 2: Variabler och datatyper. Aritmetiska uttryck. Inmatning av data.

Denna laboration är tänkt att både utföras och **redovisas** under de schemalagda timmar som finns anslagna för ändamålet. Ifall du inte har möjlighet att delta i laborationstillfällena, eller inte hinner slutföra laborationen under den schemalagda tiden kan du istället redovisa dina resultat i en **skriftlig laborationsrapport** som skickas in via Moodle. Ett obligatoriskt delkrav för att kunna bli godkänd i kursen är att **alla laborationer är godkända**.

Obs: Vissa uppgifter (markerade med en asterisk *) kommer att gås igenom gemensamt under laborationstillfällena. Dessa uppgifter godkänns alltså automatiskt om du deltar i laborationstillfället.

1. ***Modifikation av programkod***

   Utgå från additionsprogrammet nedan.

```
/* Programkod 1: Addition */

#include <stdio.h>

int main(void)
{
    int sum = 0;
    sum = 11 + 24;

    printf("11 + 24 blir %d\n", sum);
    return 0;
}
```

   Att göra: Skriv och spara programmet ovan som **`add.c`**. Kompilera och verifiera att programmet fungerar som förväntat. Skapa sedan utgående från detta program två nya program för multiplikation och division. Hitta på lämpliga testvärden. Spara dina program i filerna **`mult.c`** och **`div.c`**.

   Att redovisa: Visa dina nya program. Vad händer ifall divisionen inte går jämnt ut (pröva t.ex. att dividera 5 med 4)? Vad händer ifall du försöker dividera ett tal med noll?

2. *Heltalsvariabler*

Att göra: Modifiera **`add.c`** så att du använder dig av två heltalsvariabler **`a`** och **`b`** och beräknar värdet på **`sum`** utgående från dessa två variabler. Modifiera även **`printf()`**-satsen så att utskriften använder sig av värdet på de nya variablerna. Ifall du t.ex. ger **`a`** värdet 19 och **`b`** värdet 4 skall utskriften vara:
`19 + 4 blir 23`.

Ändrar du därefter värdet på **`a`** till t.ex. 20 skall utskriften alltså också förändras utan att du behöver ändra på **`printf()`**-raden. Testa!

Att redovisa: Visa ditt modifierade program. Vad händer om du initialiserar variabeln **`a`** till bokstaven **`'m'`** och variabeln **`b`** till bokstaven **`'n'`**? (Observera användningen av citationstecken som instruerar kompilatorn att tolka bokstäverna som **`char`**-värden!) Kan du lista ut varför resultatet blir som det blir?

3. *Flyttalsvariabler*

Att göra: Gör en ny version av **`div.c`** som använder sig av **`float`**-variabler för att lagra täljare, nämnare och resultat. Pröva att dividera 10000 med 3. Blir resultatet det förväntade?

Att redovisa: Visa ditt program. Ifall resultatet inte blev riktigt det du förväntade dig: Varför tror du att det gick som det gick? Kan programmet förbättras på något sätt?

4. *Teckenvariabler*

Att göra: Använd tillräckligt många variabler av typen **`char`** för att lagra varje bokstav i ditt förnamn. Skriv ut alla bokstäver på en enda rad genom att använda lämpligt antal **`%c`** i "formatteringssträngen" du skickar till **`printf`.**

Att redovisa: Visa ditt program.

Kommentar: Att hantera långa teckensträngar på detta sätt är en klumpig lösning - betydligt bättre vore att använda *räckor* med bokstäver. Räckor introduceras senare under kursen.

5. *Initialisering av variabler*

Att göra: Skriv ett program som deklarerar några variabler, men **inte** ger några startvärden till dessa variabler. Skriv ut värdena på dessa variabler. Vad kan du dra för slutsatser av resultatet?

Att redovisa: Dina slutsatser.

6. *Inmatning av data*

Att göra: Gör ytterligare en version av ditt additionsprogram från uppgift 2. Använd dig denna gång av **`scanf()`** för att låta användaren ange värden för de två variablerna **`a`** och **`b`**.

Att redovisa: Visa ditt program. Vad händer om du matar in t.ex. en bokstav istället för en siffra som indata till programmet?

7. (*) *Uppdatering av variabler, inmatning av data*

Att göra: Skriv ett program som först låter användaren mata in ett värde till en **`double`**-variabel, därefter fördubblar värdet på variabeln, och slutligen dividerar värdet med två. Variabelns värde skall alltså förändras under programkörningens gång. Skriv ut värdet på variabeln med tre decimalers noggrannhet.

Utskriften kan t.ex. se ut som följer:

```
Please enter a value: 5.2       <= 5.2 matas in av användaren
Let us double the value.
The value is now 10.400.
Let us divide the value by 2.
The value of is now 5.200.
```

Att redovisa: Visa ditt program.

8. (*) *Aritmetiska uttryck & användning av programbibliotek*

Att göra: Skriv ett program för att beräkna uttrycket $y = 3x^3 - 5x^2 + 6$. Använd dig av två **`double`**-variabler för att lagra värdena på **`x`** och **`y`**. Låt användaren mata in ett värde för **`x`** i början av programkörningen och skriv ut värdet på **`y`** i slutet av programkörningen.

Potensräkning (=“upphöjt i”) kan definieras som “upprepad multiplikation”.
T.ex. är $4^3 = 4 * 4 * 4 = 64$.

Lös problemet på två olika sätt:

a) Använd endast addition, multiplikation och subtraktion

b) Utnyttja funktionen **`pow()`** från C:s standardmatematikbibliotek.
Denna funktion tar två flyttal **`x`** och **`y`** som parametrar och returnerar värdet x^y
(i form av ett flyttal). Användning av matematikbiblioteket kräver att du meddelar kompilatorn att detta bibliotek måste länkas med till ditt program, vilket görs genom att ge argumentet **`-lm`** till **`gcc`**. Dessutom måste du inkludera *headerfilen* för matematikfunktionerna i ditt program genom att lägga till instruktionen **`#include <math.h>`** i början av din källkodsfil.

Att redovisa: Pröva bägge metoderna och kontrollera att de ger samma resultat för samma värde på **`x`**. Pröva även att kompilera programmet utan att länka med matematikbiblioteket – vad händer?

Tips: För att kontrollera syntaxen och få korta exempel på hur standardfunktioner används kan http://www.cplusplus.com/ varmt rekommenderas. T.ex. kan du hitta exempel på användningen av **`pow()`** här: http://www.cplusplus.com/reference/cmath/pow/