Programmering 1

Laboration 9: Pekare

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 måste du 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. ***Enkla pekare***

   Att göra: Skriv ett program som
   - deklarerar två **`double`**-variabler **`d1`** och **`d2`** och tilldelar dem värden
   - deklarerar en **`double`**-pekare **`p1`** och sätter den att peka på **`d1`**
   - skriver ut det värde som finns i den minnesadress **`p1`** hänvisar till
   - ändrar **`p1`** så att den istället pekar på **`d2`**
   - och slutligen åter skriver ut det värde som finns i den minnesadress **`p1`** hänvisar till

2. ***Pekare och funktioner***

   Att göra: Skriv en funktion **`doubleValue`** som tar en pekare till ett heltal som parameter och fördubblar detta heltals värde.

   Exempel på användning av **`doubleValue`**:

   ```
   int a = 10;
   doubleValue(&a);
   printf("a is now %d\n", a); // Bör ge resultatet 20
   ```

   Pröva också att skicka in pekare till andra datatyper till samma funktion.
   (**`short`**, **`long`**, **`float`**, **`double`**) och kontrollera resultatet.

3. ***Pekare och räckor***

   Att göra: Deklarera en heltalsräcka **`arr`** med tio element som du initialiserar till tio olika värden. Deklarera därefter en heltalspekare **`ptr`** som till en början pekar på det första elementet i räckan **`arr`**.

   Skriv ut alla värden i räckan två gånger, först på gammalt bekant sätt med hjälp av räckans index **`[0]`** – **`[9]`**, och sedan genom att använda dig av pekaren **`ptr`**.

Du kan även skriva ut minnesadressen en pekare hänvisar till i *hexadecimalt* format genom följande syntax: **`printf("%p", ptr);`**
Skriv ut minnesadresserna till alla element i räckan. Ser du något samband mellan de olika minnesadresserna?

4. **(*) *Pekare och räckor 2***

   Att göra: Utvidga barnbidragskalkylatorn från laboration 5 med en möjlighet att också beräkna finska barnbidragssummor enligt http://www.kela.fi/web/sv/barnbidrag-belopp . Lagra de åländska respektive finländska beloppen i varsin lookup-array och låt programmet använda sig av endera räckan, beroende på användarens val.

   Programmets beräkningslogik får **inte** innehålla några **if..else** satser för att avgöra vilken räcka som skall användas. Använd istället en pekare som initialiseras före beräkningen och som hänvisar till önskad räcka.

5. ***(*) Pekare och strängar***

   Att göra: Skriv en funktion **`char* contains (const char* string, char c)`** som returnerar en pekare till den första förekomsten av bokstaven **`c`** i **`string`**, eller **`NULL`** om bokstaven inte förekommer i strängen.

   (Genom att deklarerara **`char* string`** - parametern som **`const`** försäkrar vi oss om att funktionen inte kan modifiera innehållet i strängen)

   `contains` måste använda sig av pekare för att traversera strängen!

   Exempel på användning av **`contains`**:

```
char test[] = "abba";
char* result = contains(test, 'b');

if (result != NULL)
{
   // result is now pointing to the first 'b' in the string
   *result = 'x';          // replace the letter
   printf("%s\n", test);  // should give the result axba
}
```

   Obs: Vid lösning av denna uppgift kan du få kompileringsvarningar liknande dessa:
   `warning: return discards qualifiers from pointer target type`
   eller
   `warning: initialization discards qualifiers from pointer target type`
   Programmet fungerar troligen korrekt också om detta inträffar, men fundera ändå på vad dessa varningar kan tänkas innebära.

6. ***(*) Pekare och strukturer***

Att göra: Modifiera funktionen **`printStudent()`** från laboration 8 så att du skickar en pekare till en strukt som funktionsparameter. Kompilera programmet och verifiera att funktionen fortfarande fungerar korrekt.

Modifiera därefter även **`initStudent()`** så att den returnerar en pekare till en strukt. Kompilera programmet och notera eventuella varningar. Vad beror dessa på och hur kan man lösa problemet?