MÄSTARPROV 5

Tillämpning av modul 1-4

Shopping assistant

Introduktion

När man åker utomlands och handlar i en affär vill man ofta veta vad det man handlar kommer att kosta i den lokala valutan och i svenska kronor. I detta mästarprov ska en enkel shoppingassistent implementeras. Mästarprovet ska lösas enskilt.

Syfte

Syftet med mästarprovet är att du ska visa att du kan applicera den kunskap du bemästrat i modul 1-4 och sätta samman ett komplett program i enlighet med givna instruktioner.

Uppgift

Din uppgift är att skriva ett program som läser in priser i aktuell utländsk valuta och sedan presenterar summan av priserna i aktuell valuta och i svenska kronor. Användaren ska kunna ange växlingskurs.

För att se hur ditt program ska fungera (inklusive exakt vad som skrivs ut) finns det ett körbart program som du kan undersöka. Det körbara programmet nås via länk i beskrivningen till detta mästarprov.

Algoritm

Följande algoritm ska användas i programmet:

1. Så länge användaren inte valt att avsluta programmet, upprepa följande:
   - 1.1 Skriv ut menyn
   - 1.2 Läs in val
   - 1.3 Om valet är att sätta valutakurs
     - 1.3.1 Efterfråga valutakurs
     - 1.3.2 Sätt valutakurs
   - 1.4 Om valet är att konvertera priser
     - 1.4.1 Summera priser så länge användaren ej avbryter
       - 1.4.1.1 Efterfråga pris (eller klar med inmatning)
       - 1.4.1.2 Uppdatera summan (om det är ett giltigt pris)
     - 1.4.2 Skriv ut summan i lokal valuta och i svenska kronor
   - 1.5 Om valet är ogiltigt
     - 1.5.1 Skriv ut att det var ett ogiltigt val
2. Avsluta programmet

Krav på implementationen

- Funktionaliteten i steg 1.1, 1.2, 1.3.1, 1.4.1, 1.4.1.1, samt 1.4.2 ska lösas i funktioner, totalt sex funktioner förutom main-funktionen.
- Observera att funktionen som hör till steg 1.4.1.1 ska anropas från funktionen 1.4.1.
- All utskrift på skärm ska se exakt ut som enligt nedanstående exempel.

- Källkodsfilen ska heta `mp5.c`.
- I den beskrivande texten i filen ska du skriva ditt namn och din cs-användare. Vidare ska programmet och varje funktion beskrivas, se modul 2 för exempel (kodexempel funk.c).
- Programmet ska kompileras med kompilatorn `gcc` med flaggorna `-Wall` och `-std=c99`.
- Programmet ska kunna kompileras och vara körbart på institutionens linuxdatorer.
- Programmet behöver bara hantera korrekt indata. Ingen validering av indata krävs, förutom att det ska gå att använda programmet på det sätt som illustreras i nedanstående exempel från körning av programmet. Om du väljer att validera data ska valideringen vara korrekt.
- Nedan visas exempel på hur interaktionen med användaren ska se ut. Inga växlingsavgifter används. Observera att utskriften inleds och avslutas med en tom rad (tecknen i `[Tom rad]` ska ej skrivas ut).

```
[Tom rad]
Your shopping assistant

1. Set exchange rate in SEK (current rate: 1.00)
2. Convert prices from the foreign currency
3. End

Give your choice (1 - 3): 1

Give exchange rate:  9.71

1. Set exchange rate in SEK (current rate: 9.71)
2. Convert prices from the foreign currency
3. End

Give your choice (1 - 3): 4

Not a valid choice!

1. Set exchange rate in SEK (current rate: 9.71)
2. Convert prices from the foreign currency
3. End

Give your choice (1 - 3): 2

Give price (finish with < 0): 2.75
Give price (finish with < 0): 3.50
Give price (finish with < 0): -23

Sum in foreign currency: 6.25
Sum in SEK: 60.69

1. Set exchange rate in SEK (current rate: 9.71)
2. Convert prices from the foreign currency
3. End

Give your choice (1 - 3): 3

End of program!
[Tom rad]
```

Redovisning

Uppgiften redovisas genom att lämna in källkodsfilen via webbgränssnittet i Labres, se länk i beskrivningen till detta mästarprov. Din inlämnade fil kompileras och testkörs. Du kan lämna in flera gånger.

När du vill få din inlämning bedömd, gör mästarprovet för denna modul (MP5 - Mästarprov modul 5: Önskan om bedömning). I det mästarprovet ska du bara skriva din cs-användare och lämna in provet.

Tips

Här följer några tips som kan hjälpa till:

- Börja med att förstå uppgiften: Vad ska göras? Vilka krav finns det? Vad ska lämnas in?
- Skapa en fullständig förståelse för den givna algoritmen. Testkör den mot det givna exemplet och det tillhandahållna körbara programmet.
- Skapa källkodsfilen och skriv in algoritmer och beskrivande text.
- Utveckla programmet stegvis. Ett förslag är att skriva huvudfunktionen först och sedan implementera funktionerna i en ordning som är till hjälp vid utvecklandet av koden.
- Försök att testa varje steg utförligt innan nästa steg tas och kom ihåg att testa att tidigare steg fortfarande fungerar när nästa steg testas.
- Den givna algoritmen ska följas och efterfrågade funktioner ska finnas.

Kvalitetskriterier

Den inlämnade lösningen kommer att bedömas enligt följande kriterier:

Kriterium	**Godkänd**	**Godkänd med anmärkning**	**Ofullständig**
Kompilering	Utan varning	Mindre allvarlig	Allvarlig
Testkörning/ Korrekthet	Utan fel	Mindre fel	Felaktig output Räknar fel
Kommentarer	Informativa Lagom omfattning Konsekvent språk Algoritmen plus eventuella nödvändiga utökningar (steg i algoritmen kan utelämnas om bra namn på funktioner, eller annat, ger samma förståelse)	För lite eller för mycket Saknar viss information Otydliga Upprepar koden Ej konsekvent språk	Missvisande Saknar nödvändig information Saknar beskrivande text för program och/eller funktioner
Indentering	Korrekt	Något fel	Många fel Visar tecken på att ej förstå varför och hur man indenterar
Variabel-deklaration	Konsekvent Olika datatyper på olika rader Väl valda datatyper används	Olika datatyper på samma rad Blandning av deklaration i början och vid behov Mindre brister i val av datatyper	Felaktig Visar flera tecken på att ej förstå vilka datatyper som är lämpliga att använda
Namngivning (variabler, funktioner, parametrar)	Bra namn Konsekvent namngivning	Mindre bra namn Ej konsekvent namngivning	Missvisande namn Olämpliga namn
Valstrukturer	Bra struktur Bra val av villkor	Villkorsoperatorn används Allt på en rad Mindre bra val av villkor	Felaktiga villkor
Loop-strukturer	Bra val av loop-konstruktion Bra val av villkor	Mindre bra val av loop-konstruktion Mindre bra val av villkor	Felaktiga villkor `break` och `continue` används i onödan
Funktioner	Programmet är uppdelat i lämpliga funktioner med bra parametrar och returvärden	Mindre problem med val av parametrar/returvärden	Funktioner saknas Felaktiga val av parametrar/returvärden
Program-struktur	Måsvingar placeras konsekvent Måsvingar används till alla val- och loop-konstruktioner Funktionsdeklarationer och funktionsdefinitioner är konsekvent placerade	Fall av inkonsekvent placering av måsvingar Måsvingar saknas Inkonsekvent placering av funktionsdeklarationer och funktionsdefinitioner Fler än en `return`-sats `exit()` används Annat än 0 returneras	Måsvingar placeras hur som helst Onödigt många return-satser i en funktion Globala variabler Kommandona `goto` och/eller `longjmp` används
Algoritm	Följer given algoritm	Mindre avvikelse från algoritm	Följer ej algoritm