Øving 2
TDT4102

Frist: 8.2.2013

Mål for denne øvingen:

- lære grunnleggende funksjoner
- lære forskjellene mellom "call-by-value" og "call-by-reference"
- prøve å overlagre funksjoner
- lære å bruke assert-makroen
- lære å skrive koden din i flere filer

Generelle krav:

- bruk de eksakte navn og spesifikasjoner som er gitt i oppgaven
- det er valgfritt om du vil bruke en IDE (Visual Studio, Xcode), men koden må være enkel å lese, kompilere og kjøre
- alle deloppgaver skal gjøres i samme fil, unntatt når annet er spesifisert (del 7)
- alle deloppgaver skal testes fra main()

Anbefalt lesestoff:

- Kapittel 3 & 4, Absolute C++ (Walter Savitch)

Del 1: Funksjonshoder (10 poeng)

Skriv kun funksjonshodene (function headers) til de følgende funksjonene:

a) Funksjonen *hypotenuse* som tar to dobbelt presisjons flyttall (double) som argumenter, side1 og side2, og returnerer et dobbelt presisjon flyttall.

b) Funksjonen *sum* som tar inn tre heltall (integer), x, y og z, og returnerer et heltall.

c) Funksjonen *instructions* som ikke tar argumenter og ikke returnerer noe.

d) Funksjonen *intToDouble* som tar et heltall som argument og returnerer et dobbelt presisjons flytall.

Del 2: Funksjoner og prosedyreabstraksjon (10 poeng)

a) Lag funksjonen *printTime* som tar tre heltall, *hour*, *minute* og *second* som argumenter og skriver de ut til skjermen (ved å bruke *cout*).

Argumentene 3, 15 og 7 skal gi et resultat som dette:

```
3 hours, 15 minutes and 7 seconds.
```

b) Lag funksjonen *getTime* som tar et antall sekunder som argument og skriver ut det samme som funksjonen i **a)**. Du kan gjenbruke noe av koden fra forrige øving. Bruk funksjonen fra **a)** for å skrive ut på skjermen.

Del 3: Bruk av standard funksjoner (15 poeng)

I denne deloppgaven skal du lage funksjoner som bruker funksjoner fra standardbibliotekene til C++. Du må selv finne ut hvilke funksjoner du må bruke, men et bra sted å begynne å lete er i "Appendix 4" i læreboka.

a) Lag funksjonen *randomTenToFifty* som returnerer et tilfeldig heltall mellom og inkludert ti (10) og femti (50).

b) Lag funksjonene *checkTrigonometry* som tar et enkelt flyttall som argument og sjekker om

$$\tan(X) = \frac{\sin(X)}{\cos(X)}$$

stemmer for dette tallet. Returner et flyttall som gir differansen mellom $\tan(x)$ og $(\frac{\sin(x)}{\cos(x)})$. Test funksjonen din for forskjellige tall. Returnerer den 0? Bruk hva du vet om flyttall til å gi en begrunnelse om hvorfor/hvorfor ikke.

Du kan anta at funksjonen blir gitt et gyldig tall, det vil si et tall som ikke forårsaker at $\cos(x)$ blir null eller $\tan(x)$ går mot uendelig.

c) Lag funksjonen *changeCase* som tar et enkelt tegn (character) som argument og endrer det fra liten til stor bokstav eller vice versa. Med andre ord, hvis funksjonen blir gitt en liten bokstav, skal den returnere en tilsvarende stor, og hvis den blir gitt en stor skal den returnere en tilsvarende liten. Et tegn (**char**) kan i C++ deklareres som følger:

```
char myChar = 'a';
```

Del 4: Diverse funksjoner og Assert-makroen (15 poeng)

a) Skriv en funksjon som tar to tegn (**char**) som argument og returnerer **true** hvis begge er store bokstaver eller om begge er små og **false** hvis ikke, eller hvis andre tegn enn store eller små bokstaver blir gitt.

b) Skriv en funksjon som tar inn et heltall og returnerer antall siffer i tallet.

Hint: Ved bruk av heltallsdivisjon; et heltall i med n siffer, kan divideres på 10 n ganger før det blir lik null.

c) Skriv en funksjon som tar to enkelpresisjon flyttall (**float**) som argumenter og returnerer gjennomsnittet.

d) Skriv en funksjon som tester de tre forrige funksjonene med assert-makroen. Alle funksjonene skal testes flere ganger. Kall testfunksjonen din fra main(). Prøv å introdusere en feil i funksjonene dine og se hva som skjer med assert.

Hint: På grunn av flyttalls natur, bør du teste at verdien returnert fra funksjonen i c) er tilnærmet lik den forventede verdien, i stedet for nøyaktig lik. *fabs* funksjonen fra *cmath* biblioteket kan være nyttig for å klare dette.

*(**Valgfri**) The National Aeronautics and Space Administration (NASA) har etter en rekke ulykker den siste tiden bestemt seg for å prøve å gjøre noen utregninger før deres neste oppskytning. De har dog bestemt seg for å ikke gå helt bananas med forsøket sitt, og holder seg derfor nøkterne nok til å mene at en underbetalt norsk student skulle være mer enn god nok til å utføre de nødvendige utregningene. Du er tilfeldigvis denne studenten. Siden NASA er et rimelig tungrodd byråkrati, og fordi forfatteren av denne øvinga begynner å bli rimelig lat, får du en rekke funksjoner å implementere, noen av dem sågar rimelig irrelevante ved første øyekast. Du står fritt til å velge passende funksjonsnavn.*

e) *(Valgfri)* Skriv en funksjon som beregner den kinetiske energien til et objekt i bevegelse. Funksjonen skal ta objektets masse i kg og hastighet i m/s som argumenter og returnere objektets kinetiske energi.

(Den ikke relativistiske) formelen for kinetisk energi er:

$$ke = \frac{1}{2}mv^2$$

e) *(Valgfri)* Skriv en funksjon som beregner den potensielle energien til et objekt i et tyngdefelt. Funksjonen skal ta objektets masse, høyde over et tilfeldig referansepunkt og gravitasjonskonstanten som argumenter og returnere objektets potensielle energi.

(Den ikke relativistiske) formelen for potensiell energi er:

$$pe = mgh$$

e) *(Valgfri)* Skriv en funksjon som tester de to funksjonene med assert-makroen. Alle funksjonene skal testes flere ganger. Kall testfunksjonen din fra main(). Prøv å introdusere en feil i funksjonene dine og se hva som skjer med assert.

Del 5: Call-by-value / call-by-reference (20 poeng)

I denne deloppgaven skal du skrive den samme funksjonen på to ulike måter, "call-by-value" og "call-by-reference". Det anbefales at du leser om metodene i læreboken før du gjør oppgavene, slik at du er sikker du løser dem rett.

Funksjonen som skal implementeres er:

$$((A^2 + 10) * 2)$$

Hvor A er funksjonens argument.

a) Skriv funksjonen som «call-by-value.» Du skal ta inn et heltall og returnere et heltall.

b) Skriv funksjonen som «call-by-reference.» Du skal ta inn en heltallsreferanse og returnere ingenting. Resultatet av funksjonen skal lagres i argumentet du tar inn.

   Kall begge funksjonene fra main() for å se at de virker som forventet.

Del 6: Overlagring av funksjoner (15 poeng)

I denne deloppgaven skal du lage flere funksjoner med samme navn. Lag alle funksjonene i samme program og husk å teste dem fra main() for a sjekke at de fungerer som de skal.

a) Skriv funksjonen *add* som tar inn to heltall og returnerer summen som et heltall

```
Eksempel: Input: 4 , 5 - Output: 9
```

b) Skriv funksjonen *add* som tar inn to dobbelt presisjon flyttall (**double**) og returnerer summer som et dobbelt presisjon flyttall.

```
Eksempel: Input: 3.5 , 6.25 - Output: 9.75
```

c) Skriv funksjonen *add* som tar inn fire heltall som referanser (call-by-reference) og returnerer ingenting, men forandrer på de gitte referansene. Etter at funksjonen er kalt, skal det første argumentet være uforandret, det andre skal være summen av de to første argumentene, det tredje summen av de tre første og det fjerde summen av alle argumentene.

```
Eksempel:
Før funksjonskall: a=1 , b=2 , c=5 , d=1
Etter funksjonskall: a=1 , b=3 , c=8 , d=9
```

Del 7: Separate filer og "header"-filer (15 poeng)

Flytt funksjonene fra del 6 til en separat fil og link dem sammen med den originale filen din ved hjelp av en "headerfile". Test at funksjonene dine fremdeles virker ved å kalle dem fra main.

Hint: Kun "header"-filen skal inkluderes i den originale filen din og "header"-filen skal kun inneholde funksjonshodene til funksjonene dine, ikke implementasjonene. Eksempel på inkludering:

```
#include "xxx.h"
```

Del 8: Primtallsfaktorisering (Valgfri)

I denne valgfrie deloppgaven skal du implementere to metoder for å finne det største primtallet i en faktorisering av et heltall (integer) som du tar inn i funksjonen. Som i del 5 skal du bruke både "call-by-value" og "call-by-reference."

Det følgende eksemplet viser ønsket funksjonalitet:

```
Input: 42 - Output: 7
Input: 9 - Output: 3
```

a) Skriv funksjonen ved å bruke "call-by-value." Du skal ta et heltall som argument og returnere den største primfaktoren som et heltall.

b) Skriv funksjonen ved å bruke "call-by-reference." Du skal ta en heltallsreferanse som argument og returnere ingenting. Den største primfaktoren skal lagres som argumentet du tar inn.