**Tehtävä 1:** Muuta osassa 2 tehtyä **MyStandardCalculator** ohjelmaa siten, että ohjelmassa käytetään dynaamista muistinhallintaa.

## Olio-ohjelmointi: Vakiofunktio

- Jäsenfunktioiden esittelyssä luokassa voidaan jäsenfunktioon liittää sana **const**

- Tällä määrittelyllä kerrotaan kääntäjälle, että määritelty **jäsenfunktio ei muuta luokan jäsenmuuttujien arvoja**. Jos jäsenfunktioon kirjoittaa koodia, jossa jäsenmuuttujien arvoja yritetään asettaa/muuttaa, niin kääntäjä antaa tästä käännösvirheen.

- Kannattaa määritellä jäsenfunktiot vakiofunktioiksi aina kun se on mahdollista. Tällä tavalla kääntäjä voi valvoa ohjelman oikeellisuutta paremmin.
- Vakiofunktio kirjoitetaan luokkaan muodossa **void someFunction() const;** Jäsenfunktion toteutuksessa ei enää sanaa **const** kirjoiteta.

```
class MyClass
{
    public:
       void someFunction() const;
     ….
}
```

- Esimerkki vakiofunktiosta sivulla 4, yhdessä avoimen funktion esimerkin kanssa.

## Olio-ohjelmointi: Avoinfunktio

- Kääntäjä tuottaa lopulliseen suoritettavaan ohjelmaan funktioista yleensä vain yhden joukon konekielisiä komentoja.
- Kun luokan jäsenfunktiota kutsutaan, hypätään suoraan suorittamaan jäsenfunktion koodia. Kun koodi on suoritettu loppuun, palataan takaisin kutsukohtaa seuraavaan komentoon. Vaikka funktiota kutsuttaisiin monesta eri kohtaa ohjelmaa, on siitä muistissa vain yksi kopio.
- Funktioon hyppäämiseen ja sieltä palaamiseen kuluu hieman aikaa. Jos kutsuttavassa funktiossa on vain muutama rivi koodia, ohjelman toimintaa voidaan tehostaa (=suoritusnopeutta) jos funktioon siirtyminen ja sieltä palaaminen voidaan jättää pois.
- Jos luokan jäsenfunktiosta tehdään avoinfunktio, niin silloin funktion koodi kopioidaan ohjelman lopulliseen käännösversioon joka paikkaan, missä funktiota kutsutaan. Näin funktioon siirtymistä ja sieltä paluuta ei koodin tule.
- Avoimilla funktioilla on hintansa. Koska funktion komennot kopioidaan jokaiseen kutsukohtaan, suoritettavan ohjelmatiedoston koko voi kasvaa paljonkin. Nykyään tietokoneiden resurssit ovat kasvaneet niin hyviksi, että kannattaa ohjelmaa suunniteltaessa ja ohjelmoitaessa harkita, onko avoimista funktioista mitään hyötyä. ***Yleisesti näitä ei tuotteisiin menevissä koodeissa ole!***
- Koska avoimia funktioita näkee kuitenkin eri esimerkeissä käytettävän, niin seuraavalla sivulla on esimerkki miten luokan rakenteessa avoimia funktioita käytetään. Toinen vaihtoehto olisi käyttää **inline** sanaa, mutta tässä esimerkissä sitä ei käytetä.
- Avoimen funktion toteutuskoodi kirjoitetaan suoraan luokan rakenteeseen, eikä erillistä luokan **.cpp** tiedostoa tarvita.

**Tehtävä 2:**

- Tee uusi projekti nimellä **AvoinVakioStaattinen** (**Non-Qt Project->Plain C++ Application**)
- Esimerkin luonteen vuoksi luokalle ei tehdä erikseen omaa **.h** ja **.cpp** tiedostoa vaan kaikki koodi kirjoitetaan **main.cpp** tiedostoon.
- Avaa **main.cpp**, kirjoita oikealla oleva koodi tiedostoon, käy sitä läpi ja suorita ohjelmaa.
- Kun olet käynyt viereisen koodin läpi, niin lisää jäsenfunktion **showData()** viimeiseksi riviksi koodirivi **myVariable=0;** ja suorita Build toiminto.

  Kääntäjä antaa virheen, miksi?

```
#include <iostream>
using namespace std;

class MyClass
{
public:
   void setData(short paramValue) // tama on avoinfunktio, koska koodi on luokan rakenteessa
   {
      myVariable = paramValue;
   }
   void showData() const // tama on seka vakio- etta avoinfunktio
   {
       cout << "myVarible= " << myVariable << endl;
   }

private:
   short myVariable;
};

int main()
{
   MyClass objectMyClass;
   objectMyClass.setData(10);
   objectMyClass.showData();

   return 0;
}
```

HUOM! Muista tarvittaessa kääntää ohjelma käyttäen **Build** valikon vaihtoehtoa **Rebuild project…**

*Tämä siksi, koska käytetään avoimia funktioita, joiden toteutus on .h tiedostossa.*

## Olio-ohjelmointi: staattiset jäsenmuuttujat ja -funktiot

- Staattiset jäsenmuuttujat ovat yhteisiä kaikille luokasta luoduille olioille.
- Staattiseen jäsenmuuttujaan talletetaan sellaista tietoa, mitä kaikki luokan oliot yhdessä tarvitsevat.
- Kannattaa määritellä staattinen jäsenmuuttuja **private** osaan, näin sitä voidaan käyttää vain jäsenfunktioiden kautta.
  Staattinen jäsenmuuttuja esitellään luokan rakenteessa seuraavasti: **static short laskuriMuuttuja;**
- Staattiset jäsenfunktiot ovat kuten staattiset jäsenmuuttujat, ne ovat luokkakohtaisia ja yhteisiä kaikille luokan olioille. Staattisista jäsenfunktiosta on olemassa vain yksi kopio, kun taas 'normaaleista' jäsenfunktioista on olemassa yhtä monta kopioita kuin on olioita.
- Staattiset jäsenfunktiot kannattaa määritellä **public** osaan. Staattinen jäsenfunktio esitellään luokan rakenteessa seuraavasti:
  **static void jasenfunktio();**
- Staattiselle jäsenfunktiolle ei välitetä **this**-osoitinta. Tästä johtuen funktiota ei voi määritellä vakiofunktioksi.
- Koska olion jäsenmuuttujia käsitellään **this**-osoittimen avulla, staattinen jäsenfunktio ei voi käsitellä luokan tavallisia jäsenmuuttujia.
- Seuraavalla sivuilla ohjeita esimerkin **AvoinVakioStaattinen** päivittämiseen staattisen jäsenmuuttujan- ja –funktion osalta.

- Lisää luokan **private** osaan staattinen jäsenmuuttuja **static short myStaticVariable;**
- Lisää luokan **public** osaan staattinen jäsenfunktio (jäsenfunktion **showData()** jälkeen) alla olevan mukaisesti

```
static void myStaticFunction()
  {
    myStaticVariable++;
    cout << "Luokan staattisen muuttujan arvo= " << myStaticVariable << endl;
  }
```

- Jos nyt suoritat build operaation, niin kääntäjä voi antaa virheen, koska staattista jäsenmuuttujaa ei ole alustettu.
- Staattinen jäsenmuuttuja alustetaan luokan ulkopuolella. Mene **int main()** rivin yläpuolelle ja kirjoita koodirivi

```
short MyClass::myStaticVariable = 0;
```

- Lisää rivin **objectMyClass.showData();** jälkeen rivi **objectMyClass.myStaticFunction(); // kutsutaan staattista jasenfunktiota**

- Suorita build ja aja ohjelmaa.

- Kirjoita alla olevat rivit ennen **return 0;** riviä

```
MyClass objectMyClass2; // luodaan toinen olio.
objectMyClass2.myStaticFunction(); // kutsutaan staattista jasenfunktiota
```

- Suorita build ja aja ohjelmaa. Nyt huomaat miten staattisen jäsenmuuttujan arvo muuttuu.

- Staattista jäsenmuuttujaa voidaan käyttää kaikissa luokan jäsenfunktiossa, kunhan muistaa että se on luokan kaikille olioille yhteinen.

- Lisää jäsenfunktioon **setData()** koodirivi **myStaticVariable=10;**

- Suorita build ja aja ohjelmaa. Nyt huomaat miten staattisen jäsenmuuttujan arvo muuttuu.

- Normaaleja jäsenmuuttujia ei voi käyttää staattisissa jäsenfunktioissa. Jos lisäät staattiseen jäsenfunktioon rivin **myVariable = 10;** ja suorita build, niin kääntäjä antaa virheen ”invalid use of member 'MyClass::myVariable' in static member function myVariable = 10;”

- Staattisia jäsenfunktiota voidaan kutsua normaaleista jäsenfunktioista. Lisää funktioon **setData()** viimeiseksi riviksi koodirivi

  **this->myStaticFunction(); // voit kutsua funktiota myos ilman this osoitinta**

- Suorita build ja aja ohjelmaa.