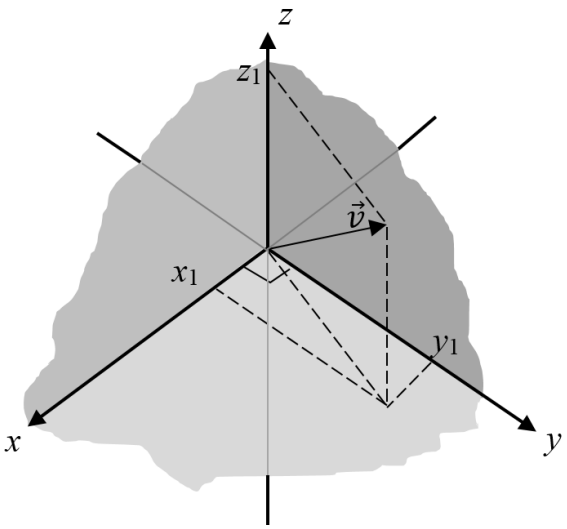
Oppgaver, 30.1.17 – PG4400 C++ progr.

Oppgave 1

En vektor er en størrelse med lengde og retning.
En vektor `v` i 3 dimensjoner (i rommet) er bestemt av 3 koordinater: $\vec{v} = [x_1, y_1, z_1]$, der x_1, y_1 og z_1 er koordinatene.
Se figuren til høyre.
Oppgaven går ut på å lage en klasse `Vector3D`.



Følgende funksjoner skal implementeres for en `Vector3D`:

- opprette en instans med koordinater lik 0.0
  ```
  Vector3D v();
  ```
- opprette en instans med gitte koordinater
  ```
  Vector3D v(float, float, float);
  ```
- beregne lengden
  ```
  float length = v.length();
  ```
- normalisere
  ```
  v.normalize();
  ```
- gi output til `cout`
  ```
  cout << v << endl;
  ```
- ta imot input fra `cin`
  ```
  cin >> v;
  ```
- likhet
  ```
  v1 == v2;
  ```
- addisjon
  ```
  v3 = v1 + v2;
  ```
- subtraksjon
  ```
  v3 = v1 - v2;
  ```
- multiplikasjon med skalar
  ```
  v2 = k · v1;
  ```
- multiplikasjon med en annen instans
  ```
  float prod = v1 · v2;
  ```

Forklaringer:

Lengden av en 3D-vektor:	$\vec{v} = [x_1, y_1, z_1] \Rightarrow$ lengden $= \sqrt{{x_1}^2 + {y_1}^2 + {z_1}^2}$
Normalisere en vektor:	hver koordinat deles på vektorens lengde
Likhet:	overlasting av operatoren ==
Input/output:	overlasting av operatorene << og >>
Multiplikasjon med en skalar:	hver koordinat ganges med skalaren – resultatet er en ny vektor overlasting av operatoren *
Multiplikasjon med en annen vektor:	hver koordinat i den ene vektoren ganges med tilsvarende koordinat i den andre og resultatene summeres – resultatet er et desimaltall overlasting av operatoren *

Oppgave 2

Lag en enkel `map` som fungerer som en dictionary (norsk/engelsk):

```
"foretrekke" er "prefer"



Innholdet i en map med norsk/engelsk:


foretrekke - prefer
liker - like
snakke - chat
trollmann - wizard
vise - show
```

Hvert element skal være et par: norsk – engelsk, der det norske ordet er `key` og det engelske er `value`.

Oppgave 3

Lag en klasse `ComponentRegister` som kan brukes til å holde rede på en bedrifts PC-utstyr. Utstyret er objekter av klassen `Component`. Denne finnes til slutt i dette dokumentet. Registeret bruker en `map<string, Component>` som en container for komponentene.

Klassen `ComponentRegister` skal minimum ha følgende funksjoner:

- `ComponentRegister();`
- `void addComponent(Component);` //legger inn en `Component` i `map`'en
- `Component getComponent(string);` //returnerer en `Component` med gitt `regNr`
- `vector<Component> getComponents();` //returnerer alle `Component`'s som en `vector`
- `int getSize();` //returnerer antall elementer i `map`'en

Lag også en `main`-funksjon som oppretter en instans av `ComponentRegister` og fyller dette med noen instanser av `Component`.

```h
#pragma once
#include<string>

using namespace std;

class Component {
public:
    Component();
    Component(string, string, int, string);
    void setRegNr(string);
    void setType(string);
    void setNumCpuCores(int);
    void setPlace(string);
    string getRegNr();
    string getType();
    int getNumCpuCores();
    string getPlace();
    string getData();
    ~Component();
private:
    string regNr;          //registreringsnummer
    string type;           //type – laptop/stasjonær
    int numCpuCores;       //antall prosessorkjerner
    string place;          //hvor komponenten er plassert
};
```

```cpp
#include "Component.h"

Component::Component() {
}

Component::Component(string rNr, string tpe, int numCC, string pl) {
    setRegNr(rNr);
    setType(tpe);
    setNumCpuCores(numCC);
    setPlace(pl);
}

void Component::setRegNr(string rNr) {
    regNr = rNr;
}

void Component::setType(string tpe) {
    type = tpe;
}

void Component::setNumCpuCores(int ncc) {
    numCpuCores = ncc;
}

void Component::setPlace(string pl) {
    place = pl;
}

string Component::getRegNr() {
    return regNr;
}

string Component::getType() {
    return type;
}

int Component::getNumCpuCores() {
    return numCpuCores;
}

string Component::getPlace() {
    return place;
}

string Component::getData() {
    return regNr + " " + type + " " + place + " " + to_string(numCpuCores);
}

Component::~Component() {
}
```