Eine Kraftwerksklasse, ihre Implementierung und Instantiierung

2. Praktikumstermin, Y-Gruppe

Praktika sind Prüfungsvorleistungen, die zum genannten Termin erbracht werden müssen. Bei Verhinderung durch Krankheit ist eine ärztliche Bescheinigung der Arbeitsunfähigkeit vorzulegen. Dieses Deckblatt ist zum Praktikumstermin ausgefüllt mitzubringen und unterschrieben abzugeben. Das Programm (C++-Projekt) wird während des Praktikums zum angegebenen Zeitpunkt im Informatik-Labor abgenommen.

Name	Vorname	Matrikelnummer	Ihre Unterschrift

Zu Beginn dieses Praktikumtermins müssen Sie Basiskenntnisse zur Thematik „Klassen und Objekte“ nachweisen und den prinzipiellen Lösungsweg erläutern können.

Ziel

Das Programm des ersten Praktikums soll umstrukturiert werden. Aus dem Strukturtyp `TKraftwerk` wird die Klasse `cKraftWerk`. Die Klasse besteht aus den Elementvariablen („Attributen“) eines Stromkraftwerks und aus den Elementfunktionen („Methoden“). Typischerweise macht man mit `public` nur die Funktionen, aber nicht die Variablen außerhalb der Klasse verfügbar. Die Funktionen sind damit die Schnittstelle der Objekte der Klasse nach außen.

cKraftwerk

```
private:
  string typ;             //Typ
  double maxP_MW;         //maximale Leistung in MW
  double aktP_MW;         //aktuelle Leistung in MW
  double kosten_EUR_kWh; //Gestehungskosten in EUR/kWh
  double CO2_g_kWh;       //CO2 im Betrieb in g/kWh
  string ort;             //Standort
public:
  cKraftwerk(string typ_, double maxP_MW_,
          double kosten_EUR_kWh_, double CO2_g_kWh_, string ort_);
  double getMaxCO2(void);
  double getAktuellCO2(void);
  void gibAus(void);
```

Es sind drei Dateien anzulegen:

- Eine .h-Datei (Header) enthält den Klassenrumpf. Der Konstruktor soll in diesem Klassenrumpf implementiert werden, also innerhalb der Schweifklammern nach `class`.
- Eine .cpp-Datei enthält die Implementierung der anderen Elementfunktionen, unter Verwendung des Klassenbezugs `cKraftwerk::<funktionsname>`.
- Die Funktion `main` steht in einer weiteren .cpp-Datei; diese inkludiert den Header der Kraftwerksklasse.

Das Programm ist als Konsolen-Anwendung mit der Entwicklungsumgebung Turbo C++ zu realisieren.

main-Funktion

In `main` sollen einige wenige Daten eines einzigen Stromkraftwerks aus einer Datei gelesen, verarbeitet und in modifizierter Form wieder ausgegeben werden:

```
int main(void)
{
  //Variablen für das Einlesen der Kraftwerksdaten:
  //Zeichenketten: typ, standort
  //Gleitkomma-Variablen: leistung, kosten, co2

  //Erstellen eines ifstream aus der Datei mit den Daten
  //Einlesen daraus: typ, leistung, kosten, co2, standort
  //Schließen des ifstream

  //Anlegen einer Objektvariablen
  //Ausgabe der Objekteigenschaften: Aufruf der Elementfunktion gibAus

  system("Pause");
  return 0;
}//main
```

Zu implementierende Elementfunktionen der Klasse cKraftwerk

Konstruktor `cKraftwerk(string typ_, double maxP_MW_, double kosten_EUR_kWh_, double CO2_g_kWh_, string ort_);`

- Die Namen der Parameter im Konstruktor sind frei wählbar. Für die Zuordnung der Werte dieser Parameter zu den entsprechenden Elementvariablen ist es aber sinnvoll, ähnliche Namen zu verwenden. Zum Beispiel kann man einfach Unterstriche an die Namen der Elementvariablen anhängen. Mit etwas Tücke kann man sogar exakt die gleichen Namen verwenden.
- Der Konstruktor setzt `aktP_MW_` auf die halbe Maximalleistung.

Elementfunktion `double getMaxCO2(void);`

- Rückgabe der CO_2-Verursachung pro Zeiteinheit bei der Maximalleistung des Kraftwerks
- Berechnung als `CO2_g_kWh*maxP_MW`
- Frage: Welche physikalische Einheit hat das Ergebnis dieser Formel?

Elementfunktion `double getAktuellCO2(void);`

- Rückgabe der CO_2-Verursachung pro Zeiteinheit bei der aktuellen Leistung des Kraftwerks
- Berechnung als `CO2_g_kWh*aktP_MW`

Elementfunktion `void gibAus(void);`

- Ausgabe von Kraftwerkstyp und Standort
- Ausgabe der aktuellen CO_2-Verursachung des Kraftwerks pro Zeiteinheit, mittels Aufruf der entsprechenden Elementfunktion

Strukturierung des Quellcodes

- Erzeugen Sie eine zusätzliche, neue Unit und speichern Sie diese in Ihrem aktuellen Quellcodeverzeichnis unter dem Namen `Kraftwerk.cpp`. Turbo C++ legt dann automatisch auch eine Datei `Kraftwerk.h` an.
- Inkludieren Sie sowohl in `Kraftwerk.cpp` wie auch in der .cpp-Datei mit `main` die Header-Dateien `<cstdlib>`, `<iostream>` und `<string>`. Öffnen Sie mittels `using namespace std;` deren Namensraum.
- Inkludieren Sie in beiden .cpp-Dateien die Header-Datei `"Kraftwerk.h"` mit Anführungszeichen statt spitzer Klammern, weil diese Datei aus Ihrem Projektverzeichnis gelesen werden soll.
- Inkludieren Sie in `"Kraftwerk.h"` die Header-Datei `<string>`. In Header-Dateien ist das `using namespace std;` ungeschickt, wie sich später zeigt. Lassen Sie das deshalb weg und schreiben Sie überall im Header `std::string` statt `string`.
- Umgeben Sie die Header-Datei mit „Include Guards“, so dass sie maximal einmal pro Compilerlauf eingebunden wird:

```
#ifndef KraftwerkH
#define KraftwerkH
... hier der eigentliche Inhalt der Header-Datei ...
#endif
```

- Verschieben Sie den alten Quellcode passend, zum Beispiel die alte `struct TKraftwerk` in die Header-Datei der neuen Unit. Passen Sie den Code dann an.
- Die Klasse `ifstream` benötigt den Header `<fstream>` und ist wie `cout` usw. im `namespace std`.

- Für die Daten legen Sie am einfachsten eine .txt-Datei an, welche die Daten mit Leerzeichen oder Zeilenumbruch getrennt enthält, zum Beispiel: `Steinkohlekraftwerk 600.0 0.035 746.0 Hintertupfingen`
- Wenn Sie diese Datei `daten.txt` nennen und in das Verzeichnis mit den .cpp-Dateien legen, können Sie sie im Programm mit "`..\\daten.txt`" und "`../daten.txt`" ansprechen.