Programmieren Übung: Template-Klassen

Klaus Kusche

Buchstaben, Worte und Wortlängen zählen

Wir wollen ein Programm schreiben, das zählt, *wie oft jeder Buchstabe* und *wie oft jedes Wort* in einer Datei vorkommt. Weiters wird gezählt, *wie oft jede Wortlänge* vorkommt.

Alle drei Zählungen sollen durch *verschiedene Instanzen derselben Template-Klasse* geschehen: Einmal zählt die Klasse Werte vom Typ **`char`** (nicht **`char *`**, sondern wirklich *einzelne* Buchstaben), einmal vom Typ **`string`** (C++ Strings, nicht C Strings), und einmal vom Typ **`int`**.
Das Template hat also den *Typ der zu zählenden Werte als Parameter*.

Hinweise zur Template-Klasse:

- Die *maximale Anzahl* **`size`** der verschiedenen Werte, die die Klasse zählen kann, soll *ebenfalls Parameter des Templates* sein: Bei den Buchstaben werden das nur 30 sein (das Hauptprogramm zählt nur die "echten" Buchstaben, keine Zahlen oder Sonderzeichen), bei den Längen sollten 50 reichen, bei den Worten sind mehr nötig (z.B. 1000).
- Die Klasse hat *3 Member*: Ein *Array vom zu zählenden Typ* mit **`size`** Elementen für die *Werte*, ein **`int`**-*Array* mit **`size`** Elementen für die dazugehörigen *Zähler* (d.h. der **`i`**-te Zähler gehört zum **`i`**-ten Wert), und einen *einzelnen* **`int`**, der angibt, *wie viele Elemente* in den beiden Arrays schon *belegt* sind.
- Der *Konstruktor* hat *keinen* Parameter und setzt nur die Anzahl der bisher belegten Elemente auf 0 (die Arrays bleiben uninitialisiert).
- Außer dem Konstruktor hat die Klasse *zwei Methoden*:
  - Eine Methode **`count`**: Sie bekommt *einen Wert übergeben* und zählt ihn:
    - Wenn der Wert schon im Werte-Array *vorkommt*, wird der dazugehörige Zähler *um eins erhöht*.
    - Wenn *nicht*, wird der Wert mit Zählerstand 1 zu beiden Arrays hinten *dazugefügt* (Achtung: Ist noch Platz? Wenn nicht: Programm mit Fehlermeldung abbrechen!)

    Die Methode hat *keinen* Returnwert.
  - Eine Methode **`print`** *ohne* Argumente und Returnwert: Sie geht die Arrays durch und gibt *jeden Wert samt Zählerstand* zeilenweise auf **`cout`** aus.

  Da beide Methoden etwas umfangreicher sind, solltest du sie nicht inline sondern *außerhalb* von **`class`** definieren!

Hinweise zum Hauptprogramm:

Das **Hauptprogramm** liegt der Angabe bei, es muss nur das Template ergänzt werden. Wenn du es selbst versuchen willst:

- Das Hauptprogramm soll eine *Datei*, deren Namen auf der Befehlszeile angegeben wird, *zeilenweise lesen*. Du kannst zu diesem Zweck das **`main`** der Musterlösung einer anderen "File zeilenweise"-Übung übernehmen, nur den Inhalt der **`while`**-Schleife musst du ändern (und die maximale Zeilenlänge sollte eventuell erhöht werden).
- Das Hauptprogramm deklariert *drei Objekte unserer Zähler-Array-Template-Klasse*, eines für **`char`** (mit 30 Elementen), eines für **`string`** (mit 1000 Elementen) und eines für **`int`** (mit 50 Elementen).
- *In* der **`while`**-Schleife sollst du die gelesene Zeile *zeichenweise* durchgehen.
    - Im ersten Schritt kümmern wir uns *nur um die Buchstaben* (jene Zeichen, für die **`isalpha(c)`** als Returnwert **`true`** liefert):

      Wir verwandeln jeden Buchstaben zuerst einmal in einen Kleinbuchstaben (**`tolower(c)`** liefert als Returnwert den zu **`c`** gehörenden Kleinbuchstaben, **`tolower`** und **`isalpha`** kommen beide aus dem Header **`cctype`**).

      Dann *zählen* wir den Buchstaben durch den Aufruf der Methode **`count`**.
    - Erst wenn dein Programm die Buchstaben richtig zählt und ausgibt, versuche *auch die Worte und deren Länge* zu zählen.
      Die Idee dazu ist Folgende:
        - Wir brauchen eine Variable, die den *Index des ersten Buchstabens* des aktuellen Wortes enthält (oder **`-1`**, wenn wir in keinem Wort sind).
        - Wenn wir an der aktuellen Position *einen Buchstaben* haben *und* bisher noch *keinen Wortanfang*, merken wir uns die *aktuelle Position* in der Eingabezeile als *Wortanfang*.
        - Wenn wir an der aktuellen Position *keinen Buchstaben*, sondern ein anderes Zeichen haben *und* uns einen Wortanfang gemerkt haben, so steht zwischen dem gemerkten Wortanfang und der aktuellen Position ein *Wort*.

          Wir *extrahieren und zählen* Wort & Länge (mit **`count`**) und setzen dann unseren gemerkten Wortanfang zurück auf "kein Wort" (**`-1`**).

          Mit folgendem Konstrukt kannst du aus einem C-String (Typ **`char []`**, so wie unsere Eingabezeile **`line`**, Header **`cstring`**) das Stück zwischen Position **`beg`** (einschließlich) und Position **`end`** (ausschließlich) *ausschneiden* und in einen neuen C++-String (Typ **`string`**, Header **`string`**) verwandeln:

          ```
          string(&(line[beg]), end - beg)
          ```

          Tipp: Mach *auch für die Endemarkierung* **`'\0'`** noch einen Schleifenumlauf, damit das *letzte Wort* der Zeile noch gezählt wird!
- *Nach* der **`while`**-Schleife wird für die drei Arrays die **`print`**-Methode aufgerufen.

Zusatzaufgaben:

- Die Lösung mit zwei getrennten Member-Arrays für Werte und deren Zähler ist nicht besonders elegant.

  Kannst du eine *zweite Template-Klasse für Paare* von einem Wert **`val`** (von beliebigem Typ) und dem dazugehörigen **`int`**-Zähler **`cnt`** einführen und in der Zähler-Klasse dann *ein einziges Array solcher Paare* verwenden?

  Die Paar-Klasse braucht *keine Methoden und keinen Konstruktor*:
  Die beiden Member dürfen **`public`** sein und direkt angesprochen werden, es reicht daher eine normale **`struct`** statt **`class`**.

- Kannst du das *Einfügen* neuer Werte in der Methode **`count`** so umbauen, dass die Werte im Array stets *sortiert* sind (damit die Ausgabe der Zähler sortiert nach Werten ist)?

  Erinnere dich an das übliche Vorgehen: Array von *hinten nach vorne* durchlaufen und bestehende Elemente *um einen Platz nach hinten verschieben* (in *beiden* Arrays!), bis an der richtigen Stelle ein Loch für das neue Element frei ist (die richtige Stelle kann man sich gleich beim Suchen des Wertes merken, dann braucht man beim Verschieben nicht mehr die Werte der Elemente vergleichen!).

- Kannst du statt der Methode **`print`** eine globale Template-Funktion **`operator<<`** definieren, die dasselbe leistet (und den Aufruf im **`main`** entsprechend ändern)?

  `operator<<` soll ein **`friend`** unserer Zähler-Template-Klasse sein.