Praktikumstermin Nr. 06, INF: HTML Datei aus Vorlage & Daten

Abgabe im GIP-INF Praktikum der Woche 14.-18.11.2022.

(Pflicht-) Aufgabe INF-06.01: Webseitendatei erstellen aus Vorlagendatei und Datendatei (Dateioperationen, `struct`)

Hinweis:
Sie brauchen im Praktikum nur das resultierende Programm vorzuzeigen, welches letztendlich alle Anforderungen der kompletten Aufgabe löst. Einzelne Zwischenversionen (z.B. gemäß den unten angegebenen Arbeitsschritten, falls Sie sich an diesen orientieren wollen) brauchen Sie nicht dauerhaft zu speichern bzw. im Praktikum vorzuzeigen.

Legen Sie in VS Code ein neues leeres C++ Projekt für diese Praktikumsaufgabe an. Den Namen des Projekts können Sie frei wählen.

Öffnen Sie nun den Ordner mit den Dateien dieses Projekts im Windows Explorer: Klicken Sie dazu mit der rechten Maustaste in VS Code auf den Projektnamen, dann *"Im Datei-Explorer anzeigen"* auswählen.

Laden Sie folgende zwei Dateien aus dem GIP Praktikumsordner in Ilias in das im Datei-Explorer geöffnete Verzeichnis des Projekts herunter:

`webseite.html.tmpl`, die Webseiten-Vorlagendatei (*„Vorlage" == „Template"*)
Achtung: Diese Datei heißt nach dem Herunterladen aus Ilias aus Sicherheitsgründen leider `webseitehtmltmpl.sec`. Sie müssen die heruntergeladene Datei also umbenennen in `webseite.html.tmpl`

`personendaten.txt`, die Datendatei.

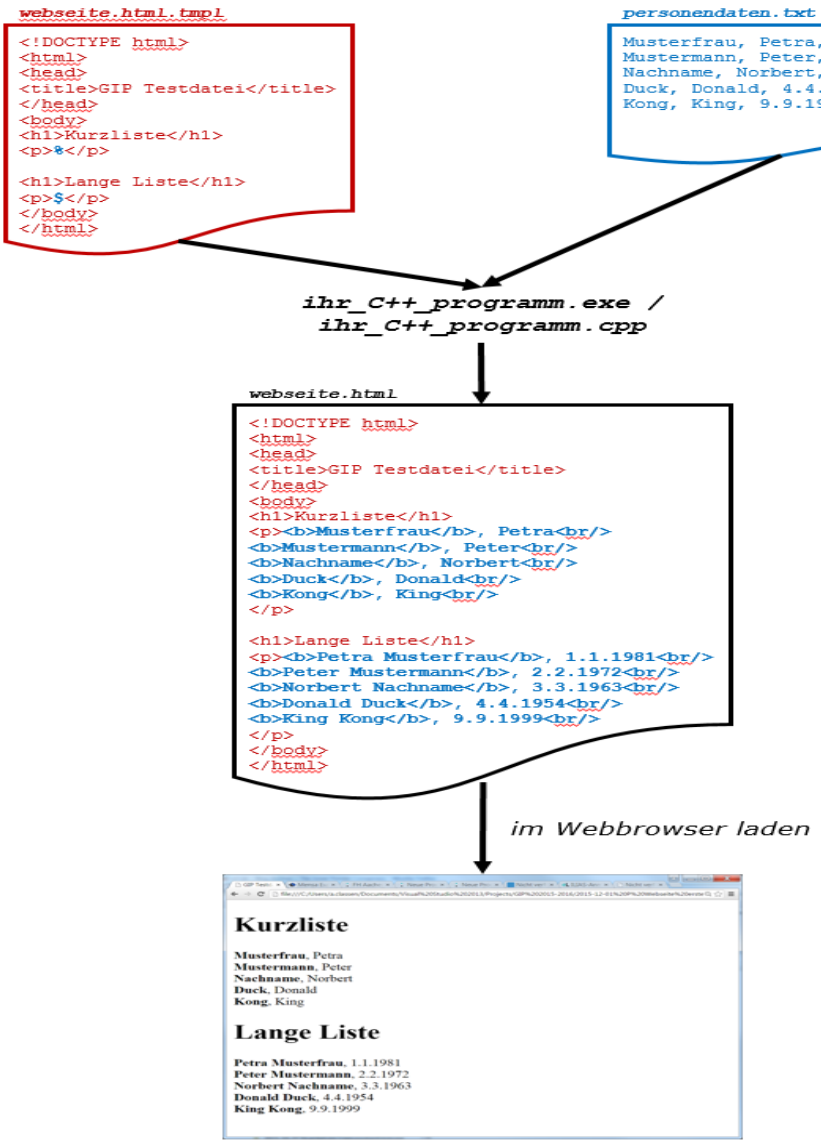
Gesamtaufgabe:
Schreiben Sie ein C++ Programm, welches die beiden Dateien einliest und mittels der Daten aus den beiden Dateien eine neue Ausgabedatei `webseite.html` schreibt, gemäß den unten beschriebenen Regeln.

Beim Inhalt der Ausgabedatei `webseite.html` wird es sich um eine Webseitendatei im HTML Format handeln. Öffnen Sie daher später zur Überprüfung, ob Ihr C++ Programm richtig gearbeitet hat, die Ausgabedatei in einem Webbrowser (z.B. in Firefox per Tastendruck `Strg-O`), um sie entsprechend aufbereitet anzeigen zu lassen.
Sollten Sie Ihr C++ Programm erneut starten, so können Sie den

aktualisierten Inhalt der `webseite.html` Datei mittels *`Reload`* im Webbrowser neu anzeigen lassen.

Das Vorgehen Ihres C++ Programms bei der Erzeugung der Ausgabedatei `webseite.html` sei dabei wie folgt:

- Im Prinzip wird der Inhalt der Vorlagendatei `webseite.html.tmpl` gelesen und in großen Teilen unverändert in die Ausgabedatei `webseite.html` geschrieben.
- Kommt in einer Zeile der Vorlagendatei das Zeichen `%` vor, so soll dieses einzelne Zeichen ersetzt werden durch folgenden längeren Text: *Den Text für die Kurz-Liste der Personen.*
  Jeder „kurze" Personeneintrag der Liste besteht aus dem Nachnamen (aus der Datendatei `personendaten.txt`) in Fettschrift, gefolgt von einem Komma und einem Leerzeichen, gefolgt vom Vornamen in normaler Schrift.
  Wie dieser Text für den Inhalt der Liste genau gebildet wird, ist weiter unten in dieser Praktikumsanleitung beschrieben.
- Kommt in einer Zeile der Vorlagendatei das Zeichen **`$`** vor, so soll dieses Zeichen ersetzt werden durch folgenden Text:
  Den Text für die „Langform-Liste". Jeder „lange Personeneintrag" besteht aus dem Vornamen gefolgt vom Nachnamen, beides in Fettschrift, gefolgt von einem Komma und einem Leerzeichen, gefolgt vom Geburtsdatum. Außer dem Vor- und dem Nachnamen sei der ganze Text in normaler Schrift.
  Wie der Text für diese Liste genau gebildet wird, ist weiter unten beschrieben.



Die Listentexte (siehe blaue Sektionen in `webseite.html` im Diagramm) ergeben sich wie folgt:

- Jede Listenzeile entspricht einer Datenzeile der Datendatei.
- Jede Listenzeile wird abgeschlossen mit dem Text `<br/>`.
- Der Text der Listenzeile ergibt sich aus den Zeilen der Datendatei, die so wie in der Aufgabenstellung oben beschrieben umgewandelt werden.
- Ein Text wird in Fettdruck ausgegeben, wenn er innerhalb der Zeichen `<b>...</b>` steht (in HTML als "tags" bezeichnet) .

Testlauf:

Keine sichtbaren Eingaben und Ausgaben des Programms. Reines Lesen und Schreiben der Dateien. Ausgabe im Webbrowser sichtbar.

Ihre möglichen Arbeitsschritte (sie müssen diese Schritte nicht unbedingt befolgen, dies ist nur ein "Vorschlag"; ihr Programm muss auch nicht unbedingt den hier gemachten Codestruktur-Vorschlägen entsprechen):

1. Schreiben Sie im ersten vorgeschlagenen Arbeitsschritt ein C++ Programm, welches die Datendatei `personendaten.txt` zeilenweise einliest und jede eingelesene Zeile wieder auf den Bildschirm ausgibt.
Pseudo-Code:

```
#include <iostream>
#include <string>
#include <fstream>
// Weglassen / vermeiden: using namespace std;

int main()
{
    std::string eingabezeile;

    Datendatei "personendaten.txt" als textuelle Eingabedatei öffnen;

    solange(eingabezeile aus Datendatei lesen) {
        eingabezeile ausgeben;
    }

    Datendatei schließen;

    return 0;
}
```

2a. Erweitern Sie dieses C++ Programm:

- Definieren Sie in einer separaten Headerdatei `person.h` einen strukturierten Datentyp für Personen.

- Definieren Sie in einer separaten Implementierungsdatei `person.cpp` eine Funktion …
  `Person extrahiere_person(std::string eingabezeile)`
  … welche aus einer Eingabezeile die Personendaten extrahiert:

Die Eingabezeile soll am ersten Komma aufgespalten werden und der Teil bis zum Komma als Nachname verwendet werden.

Der zweite Teil der Aufspaltung soll wieder am ersten Komma aufgespalten werden. Der Teil bis zum Komma soll als Vorname benutzt werden, der Teil dahinter als Geburtsdatum.

Benutzen Sie zum Aufspalten die Funktion `spalte_ab_erstem()` aus den Offline-Aufgaben. Platzieren Sie diese Funktion in separaten Dateien `texte.h` und `texte.cpp`.

Die Funktion `extrahiere_person()` soll dann den so erstellten `Person`-Wert zurückgeben.

Lassen Sie von Ihrem C++Hauptprogramm die Daten der erhaltenen Person ausgeben. *Pseudo-Code:*

```
// In person.h: ////////////////////////
struct Person { ... };

// In person.cpp: ////////////////////////
Person extrahiere_person(std::string eingabezeile)
{
    Person p;
    std::string rest;
    spalte_ab_erstem(eingabezeile, ',', p.nachname, rest);
    spalte_ab_erstem(rest, ',', p.vorname, p.geburtsdatum);
    return p;
}

// Datei main.cpp: ////////////////////////
#include <iostream>
#include <string>
#include <fstream>
// ... ggfs. weitere include ...

int main()
{
    std::string eingabezeile = "";

    Datendatei "personendaten.txt" als textuelle Eingabedatei öffnen;

    solange(eingabezeile aus Datendatei lesen) {
        Person person = extrahiere_person(eingabezeile);
        // Dann person.vorname, person.nachname,
        //      person.geburtsdatum ausgeben
    }

    Datendatei schließen;

    return 0;
}
```

2b. Fällt Ihnen auf, dass die einzelnen Personendaten noch unnötige Leerzeichen am Anfang und Ende der Einzelwerte der Person enthalten können?

Erweitern Sie die Funktion `extrahiere_person()`, so dass bei der Rückgabe des Werts `p` die Komponentenwerte `vorname`, `nachname`, `geburtsdatum` "gesäubert" werden. Benutzen Sie dazu die `std::string trimme(std::string s)` Funktion aus den Offline-Aufgaben (Code der Funktion so ändern, dass jetzt Leerzeichen statt Pluszeichen entfernt werden), die Sie ebenfalls in `texte.h` bzw. `texte.cpp` platzieren:

```
Person extrahiere_person(std::string eingabezeile)
{
    Person p; std::string rest = "";
    spalte_ab_erstem(eingabezeile, ',', p.nachname, rest);
    spalte_ab_erstem(rest, ',', p.vorname, p.geburtsdatum);
    p.nachname = trimme(p.nachname);
    p.vorname = trimme(p.vorname);
    p.geburtsdatum = trimme(p.geburtsdatum);
    return p;
}
```

3. Erweitern Sie nun Ihr Hauptprogramm, um den Kurztext gemäß den Vorgaben erstellen zu lassen. Die bisherige Ausgabe der Personenwerte kann wieder weggelassen werden.

```
int main()
{
    std::string eingabezeile = "", kurztext = "", langtext = "";

    Datendatei "personendaten.txt" als textuelle Eingabedatei öffnen;

    solange(eingabezeile aus Datendatei lesen) {
        Person person = extrahiere_person(eingabezeile);
        // Kurztext erstellen ...
        kurztext += br(
                        b(person.nachname) + ", " + person.vorname
                    ) + "\n";
    }

    Datendatei schließen;

    return 0;
}
```

Es werden dabei zwei Hilfsfunktionen benötigt:
`std::string br(std::string s)` hängt an den String `s` den Text `<br/>` an.
`std::string b(std::string s)` hängt vor den String `s` den Text `<b>` und dahinter den Text `</b>`.
Diese beiden Funktionen können Sie in der `main.cpp` platzieren.

4. Erweitern Sie nun Ihr Hauptprogramm, um den Langtext gemäß den Vorgaben erstellen zu lassen.

```
int main()
{
    std::string eingabezeile = "", kurztext = "", langtext = "";

    Datendatei "personendaten.txt" als textuelle Eingabedatei öffnen;

    solange(eingabezeile aus Datendatei lesen) {
        Person person = extrahiere_person(eingabezeile);
        // Kurztext erstellen ...
        kurztext += br(
                        b(person.nachname) + ", " + person.vorname
                    ) + "\n";
        // Langtext erstellen ...
        langtext += br(
                        b(person.vorname + " " + person.nachname) +
                        ", " +
                        person.geburtsdatum
                    ) + "\n";
    }

    Datendatei schließen;

    return 0;
}
```

5. Erweitern Sie Ihr Hauptprogramm, um das Lesen der Vorlagendatei und das Schreiben der Ausgabedatei. Benutzen Sie dabei die `ersetze()` Funktion aus den Offline-Aufgaben.

```
int main()
{
    std::string eingabezeile = "", kurztext = "", langtext = "";

    Datendatei "personendaten.txt" als textuelle Eingabedatei öffnen;

    solange(eingabezeile aus Datendatei lesen) {
        Person person = extrahiere_person(eingabezeile);
        // Kurztext erstellen ...
        kurztext += br(
                        b(person.nachname) + ", " + person.vorname
                    ) + "\n";
        // Langtext erstellen ...
        langtext += br(
                        b(person.vorname + " " + person.nachname) +
                        ", " +
                        person.geburtsdatum
                    ) + "\n";
    }

    Datendatei schließen;

    Templatedatei "webseite.html.tmpl" als textuelle Eingabedatei öffnen;

    Ausgabedatei "webseite.html" als textuelle Datei zum Schreiben öffnen;

    solange(eingabezeile aus Templatedatei lesen) {
        eingabezeile = ersetze(eingabezeile, '%', kurztext);
        eingabezeile = ersetze(eingabezeile, '$', langtext);

        Schreibe eingabezeile + "\n" in die Ausgabedatei;
    }

    Templatedatei schließen;

    Ausgabedatei schließen;

    return 0;
}
```