Praktikumstermin Nr. 08, INF: HTML Datei aus Template & Daten, Spiel

Aufgabe INF-08.01: Webseitendatei erstellen aus Vorlagendatei und Datendatei (Dateioperationen, Stringoperationen)

Hinweis:
Sie brauchen im Praktikum nur das resultierende Programm vorzuzeigen, welches 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 Visual Studio ein neues leeres C++ Projekt für diese Praktikumsaufgabe an. Den Namen des Projekts können Sie frei wählen. Legen Sie dann innerhalb des Projekts eine neue C++ Quelltextdatei (`.cpp` Datei) an. Auch hier können Sie den Namen der Datei frei wählen (Endung `.cpp`).

Öffnen Sie nun den Ordner mit den Dateien dieses Projekts im Windows Explorer: Klicken Sie dazu mit der rechten Maustaste in Visual Studio auf den Projektnamen, dann *"Ordner in Datei-Explorer öffnen"* (zweiter Eintrag von unten in dem Kontextmenü des Projekts) auswählen.

Laden Sie folgende Dateien aus dem GIP Praktikumsordner in Ilias herunter (erst einmal in einen beliebigen Zielordner):

`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.

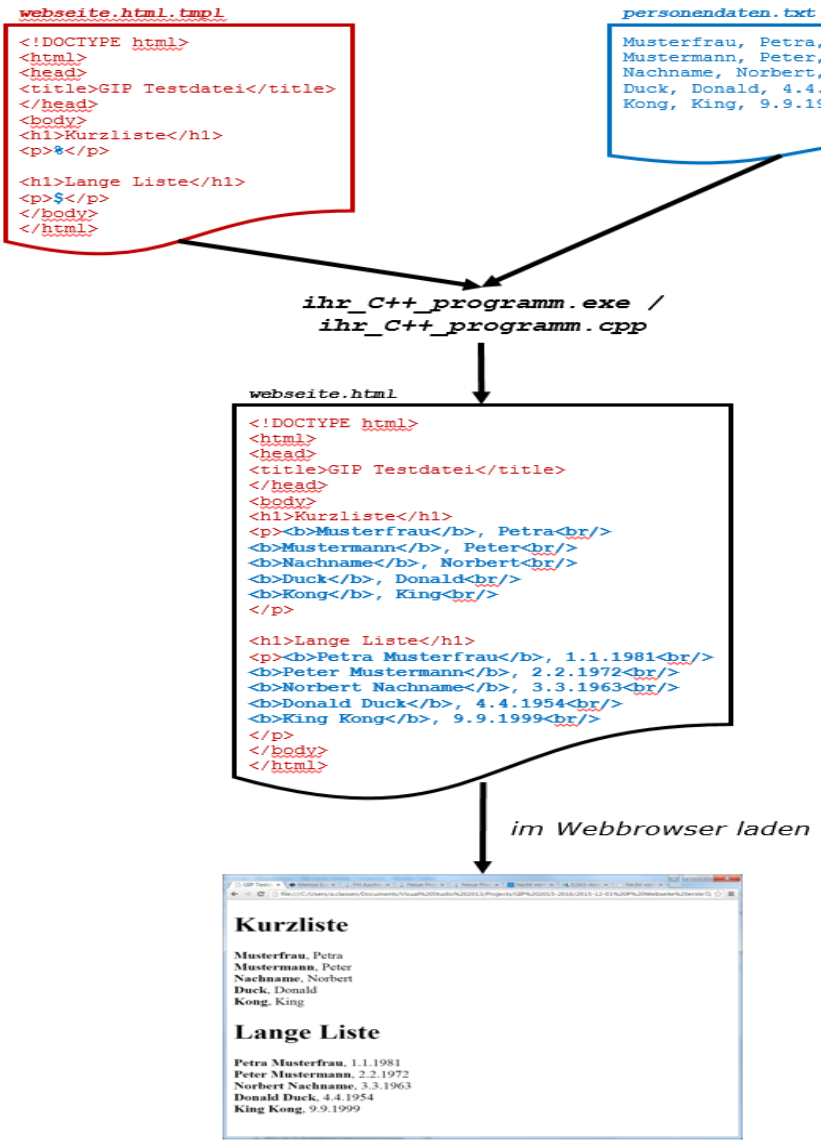
Kopieren oder verschieben Sie diese Dateien in den Ordner mit den Dateien Ihres Projekts.

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 Webseite im HTML Format handeln. Öffnen Sie zur Überprüfung, ob Ihr 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 Datei mittels "*Reload*" im Webbrowser neu anzeigen lassen.

Das Vorgehen Ihres C++ Programms bei der Erzeugung der Ausgabedatei sei dabei wie folgt:

- Im Prinzip wird der Inhalt der Vorlagendatei gelesen und meist unverändert in die Ausgabedatei 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) 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 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 wie in der Aufgabenstellung oben beschrieben umgewandelt werden.
- Ein Text wird in Fettdruck ausgegeben, wenn er innerhalb die Zeichen `<b>...</b>` steht.

Testlauf:

Keine sichtbaren Eingaben und Ausgaben des Programms. Reines Lesen und Schreiben der Dateien.

Arbeitsschritte (Sie müssen diese Schritte nicht unbedingt befolgen, dies ist nur ein "Angebot"; ihr Programm muss auch nicht unbedingt den hier gemachten Vorschlägen entsprechen):

1. Schreiben Sie 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>
using namespace std;

int main()
{
    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(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 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(string eingabezeile)
{
    Person p;
    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>
// ... weitere include ...
using namespace std;

int main()
{
    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?

Falls ja: 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 `string trimme(string s)` Funktion aus den Offline-Aufgaben (Code der Funktion so ändern, dass statt Pluszeichen jetzt Leerzeichen entfernt werden), dir Sie ebenfalls in `texte.h` bzw. `texte.cpp` platzieren:

```
Person extrahiere_person(string eingabezeile)
{
    Person p; 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()
{
    string eingabezeile = "", kurztext = "", langtext = "";
    Datendatei "personendaten.txt" als textuelle Eingabedatei öffnen;
    solange(eingabezeile aus Datendatei lesen) {
        Person person = extrahiere_person(eingabezeile);
        // kurztext ausgeben lassen ...
        kurztext += br(
                        b(person.nachname) + ", " + person.vorname
                    ) + "\n";
    }
    Datendatei schließen;
    return 0;
}
```

Es werden dabei zwei Hilfsfunktionen benötigt:
`string br(string s)` hängt an den String `s` den Text `<br/>` an.
`string b(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()
{
    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()
{
    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;
}
```

Aufgabe INF-08.02: Spiel „Regnende Kästchen“

Programmieren Sie ein einfaches „Computerspiel“ gemäß den folgenden Anforderungen, unter Benutzung der `CImg` Library.

Bei dem Spiel „regnen“ Kästchen von oben nach unten. Durch Anklicken der Kästchen werden diese wieder nach oben zurückgesetzt und regnen dann mit leicht erhöhter Geschwindigkeit erneut nieder. Das Spiel endet, wenn ein Kästchen die untere Kante des Fensters überschritten hat.

Achten Sie bei Ihrer Lösung darauf, dass die Kästchen auch wirklich an den oberen Rand des Fensters zurückgesetzt werden. D.h. die Oberkante des Kästchens sollte beim Zurücksetzen an den oberen Rand des Fensters stoßen, es sollte zuerst einmal keine Lücke sichtbar sein, bevor das Kästchen erneut „herunterfällt".

Legen Sie ein neues leeres Projekt an. Legen Sie dort eine neue Headerdatei `CImgGIP05.h` an und copy-pasten Sie den Inhalt der entsprechenden Datei aus Ilias in diese Datei. Legen Sie ferner eine Quellcodedatei (`.cpp` Datei) mit einem von Ihnen frei wählbaren Namen an und copy-pasten Sie den gegebenen Coderahmen in diese Datei. Komplettieren Sie dann den Coderahmen.

```
#include <iostream>
#define CIMGGIP_MAIN
#include "CImgGIP05.h"
using namespace std;

struct Box
{
    int x;
    int y;
    int delta_y; // aktuelle Fallgeschwindigkeit dieses Kaestchens
};

const int box_max = 10, box_size = 40;

// draw_boxes():
// Die Anzahl der Kaestchen wird nicht als zweiter Parameter uebergeben,
// da diese Anzahl als globale Konstante box_max im gesamten Programm bekannt ist ...
void draw_boxes(Box boxes[])
{
    // Loesche den bisherigen Fensterinhalt komplett,
    // d.h. komplettes Neuzeichnen aller Kaestchen etc ...
    gip_white_background();

    // Und jetzt alle Kaestchen zeichnen mittels gip_draw_rectangle().
    // Linke obere Ecke: (boxes[i].x, boxes[i].y)
    // Groesse: box_size x box_size
    // Farbe: blue

    // TO DO

}

// update_boxes():
// 1. Berechne neue y-Koordinate (jedes Kaestchen boxes[i] faellt um boxes[i].delta_y)
// 2. Pruefe, ob ein Kaestchen das untere Fensterende gip_win_size_y (ist ein int)
//    ueberschritten hat. Falls ja: gib sofort false zurueck
// 3. (Sonst:) gib true zurueck, wenn keines der Kaestchen den unteren Rand
//             ueberschritten hatte
// Die Anzahl der Kaestchen wird nicht als zweiter Parameter uebergeben,
// da diese Anzahl als globale Konstante box_max im gesamten Programm bekannt ist ...
bool update_boxes(Box boxes[])
```

```
{
    // TO DO
}

// handle_mouse_click():
// Annahme: Funktion wird nur aufgerufen, wenn die Maus wirklich
// geklickt wurde. Die Funktion braucht dies also nicht mehr zu pruefen.
//
// Nimm die Koordinaten der Mausposition und pruefe dann fuer jedes Kaestchen,
// ob die Koordinaten innerhalb des Kaestchens liegen.
// Falls ja:
// * Erhoehe die Fallgeschwindigkeit des Kaestchens um 10
// * Setze das Kaestchen an den oberen Rand
void handle_mouse_click(Box boxes[])
{
    int mouse_x = gip_mouse_x();
    int mouse_y = gip_mouse_y();

    // TO DO
}

int main()
{
    Box boxes[box_max] = { 0 };
    bool keep_going = true;

    // Initialisiere alle Kaestchen ...
    for (int i = 0; i < box_max; i++)
    {
        // Die "+10" bewirken, dass alle Kaestchen leicht nach rechts versetzt werden
        // und somit links vom linksten Kaestchen noch etwas Platz bleibt (dieses
        // also nicht am Rand klebt).
        // Die "+20" sorgen fuer einen Zwischenraum von 20 zwischen den Kaestchen ...
        boxes[i].x = i * (box_size + 20) + 10;
        boxes[i].y = 0;
        // Startgeschwindigkeit ist ganzzahlige Zufallszahl zwischen 10 und 30 ...
        boxes[i].delta_y = gip_random(10, 30);
    }

    while (keep_going && gip_window_not_closed())
    {
        draw_boxes(boxes);

        for (int loop_count = 0; loop_count < 200; loop_count++)
        {
            gip_wait(5); // warte 5 Milli-Sekunden
            if (gip_mouse_button1_pressed())
            {
                handle_mouse_click(boxes);
            }
        }
        // Berechne die neue Fall-Position aller Kaestchen und pruefe,
        // ob eines der Kaestchen unten aus dem Fenster "herausgefallen" ist
        // (falls ja, wird der Wert false zurueckgegeben) ...
        keep_going = update_boxes(boxes);
    }

    return 0;
```

```
}
```

Testlauf: