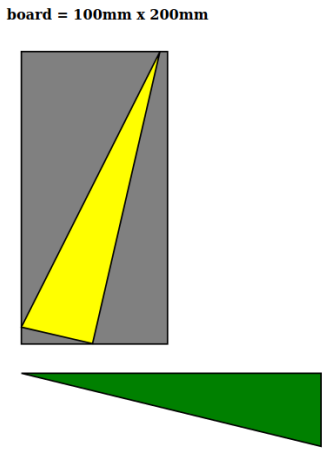
# SNP: User-defined Typen, Funktionsparameter by-value


**Wochentag eines Datums**




**Einpassen eines Dreiecks**



## 1 Übersicht

In der ersten Aufgabe berechnen Sie den Wochentag eines beliebigen Datums unseres Kalenders. Damit machen Sie sich mit der anwender-definierten Typen (**`enum`**, **`struct`**) vertraut, und Sie schreiben verschiedene Hilfsfunktionen wir **`is_leap_year`**, **`is_valid_date`**, etc.

In der zweiten Aufgabe vertiefen Sie weiter obige Themen indem Sie gegebenes Dreieck versuchen auf einem Brett anzuordnen, so dass es ausgesägt werden kann. Dazu definieren Sie Typen um einen zweidimensionalen Punkt, ein Dreieck und ein waagrechtes Rechteck speichern zu können. Sie schreiben Funktionen welche für ein Dreieck das umschliessende horizontale Rechteck (Bounding-Box) berechnet. Schliesslich schreiben Sie Funktionen um Koordinaten und Rechtecke zu vergleichen.

## 2 Lernziele

In diesem Praktikum lernen Sie **`struct`** und **`enum`** anzuwenden und Funktionen zu definieren und aufzurufen.

- Sie können **`enum`** Typen definieren und deren Werte bestimmen und abfragen.
- Sie können **`struct`** Typen definieren, initialisieren und auf die Elemente zugreifen.
- Sie können ein Programm schreiben welches aus mehreren Funktionen besteht.
- Sie können Funktionen deklarieren, definieren und aufrufen.
- Sie können Command-Line Argumente mittels **`sscanf`** parsen und die Eingabe auf Fehler überprüfen.

Die Bewertung dieses Praktikums ist am Ende angegeben.

Erweitern Sie die vorgegebenen Code Gerüste, welche im **`git`** Repository **`snp-lab-code`** verfügbar sind.

# 3 Aufgabe 1: Wochentag eines Datums

Erweitern Sie das vorgegebenen Programm Gerüst an den bezeichneten Stellen so, dass das Programm von der Kommando Zeile ein Argument entgegennimmt, es auf Gültigkeit überprüft und schliesslich den Wochentag für das gegebene Datum berechnet und ausgibt.

Prüfen Sie die Umsetzung beider Teilaufgaben mittels **`make test`**.

## 3.1 Teilaufgabe Argumente Parsen und auf Korrektheit prüfen

Das Argument stellt ein gültiges Datum unseres Gregorianischen Kalenders dar (d.h. ein Datum ab Donnerstag, den 15. Oktober 1582, mit der Gregorianischen Schaltjahr Regel).

Wenn kein Argument gegeben ist oder wenn das eingegebene Datum nicht gültig ist, soll das Programm einem Hilfetext auf **`stderr`** ausgeben und mit **`EXIT_FAILURE`** Exit Code terminieren. Wenn ein gültiges Datum erkannt wurde terminiert das Programm mit Exit Code **`EXIT_SUCCESS`**.

### Argument Format

Das Format des Kommando Zeilen Arguments soll ***`yyyy-mm-dd`*** sein, wobei ***`yyyy`*** für das vierstellige Jahr, ***`mm`*** für einen 1-2-stelligen Monat (1…12) und ***`dd`*** für einen Tag des Monats, beginnend mit 01. Z.B. **`2020-02-29`**.

### Korrektes Datum

Das Datum muss alle folgenden Bedingungen erfüllen damit es als korrekt erkannt wird:

1. Obergrenze für ein «sinnvolles» Datum ist das Jahr 9999
2. es muss Gregorianisch sein, d.h. ab 15. Oktober 1582 (inklusive)
3. es darf nur Monate von 1 für Januar bis 12 für Dezember beinhalten
4. der Tag muss grösser oder gleich 1 sein
5. der Tag darf nicht grösser als 31 sein für Monate mit einer Länge von 31 Tagen
6. der Tag darf nicht grösser als 30 sein für Monate mit einer Länge von 30 Tagen
7. der Tag darf für den Februar nicht grösser sein als 29 für ein Schaltjahr
8. der Tag darf für den Februar nicht grösser sein als 28 für ein Nicht-Schaltjahr

Bedingung für ein Schaltjahr:

1. wenn das Jahr durch 400 ohne Rest teilbar ist, **ist** es ein Schaltjahr
2. sonst, wenn das Jahr durch 100 ohne Rest teilbar ist, ist es **kein** Schaltjahr

3. sonst, wenn das Jahr durch 4 ohne Rest teilbar ist, **ist** es ein Schaltjahr
4. sonst ist es **kein** Schaltjahr

**Vorgaben an die Umsetzung**

1. Definieren Sie einen **`enum`** Typen mit (**`typedef`**) Namen **`month_t`** dessen Werte die Englischen 3-Zeichen Abkürzungen der Monate sind, nämlich **`Jan`**, **`Feb`**, … **`Dec`** und stellen Sie sicher dass die Abkürzungen für die uns geläufigen Monatsnummer stehen.
2. Definierend Sie einen **`struct`** Typen mit (**`typedef`**) Namen **`date_t`** und den **`int`** Elementen **`year`**, **`month`**, **`day`**. Lesen Sie das Argument (falls vorhanden) via **`sscanf`** und dem Formatstring **`"%d-%d-%d"`** in die drei Elemente einer **`Date`** Variable. Siehe dazu die Hinweise im Anhang.
3. Für die Berechnung der Monatslänge implementieren Sie die Hilfsfunktion **`is_leap_year(date_t date)`** (nach obigen Vorgaben). Der Return Wert **`0`** bedeutet «Kein Schaltjahr», **`1`** bedeutet «Schaltjahr».
4. Implementieren Sie die Funktion **`int get_month_length(date_t date)`**. Diese soll für den Monat des Datums die Monatslänge (was dem letzten Tag des Monats entspricht) ausgeben – geben Sie 0 für ungültige Monatswerte zurück.
5. Schliesslich implementieren Sie die Funktion **`int is_gregorian_date(date_t date)`** welche prüft, ob ein gegebenes Datum im Bereich 15. Oktober 1582 und dem Jahr 9999 ist (**`0`** = nein, **`1`** = ja).
6. Implementieren Sie eine Funktion **`int is_valid_date(date_t date)`**, welche obige Bedingungen für ein gültiges Datum umsetzt. Der Return Wert **`0`** bedeutet «Kein gültiges Datum», **`1`** bedeutet «Gültiges Datum». Benutzen Sie für die Prüfung des Datums die **`month_t`** Werte wo immer möglich und sinnvoll. Verwenden Sie die oben implementierten Hilfsfunktionen.

**Hinweise**

- Beachten Sie die Kommentare im Code für die geforderten Implementierungs-Details.

## 3.2 Teilaufgabe Wochentag Berechnung

Schreiben Sie eine Funktion welche zu einem Datum den Wochentag berechnet.

Die Formel wird Georg Glaeser zugeschrieben, möglicherweise angelehnt an eine Formel von Carl Friedrich Gauss.

$$w = (d + \lfloor 2{,}6 \cdot m - 0{,}2 \rfloor + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c) \bmod 7$$

(Quelle: https://de.wikipedia.org/wiki/Wochentagsberechnung)

Hier ist eine für C abgewandelte Variante davon.

```
weekday = ((day + (13 * m - 1) / 5 + y + y / 4 + c / 4 - 2 * c) % 7 + 7) % 7
```

alle Zahlen sind **`int`** Werte und alles basiert auf **`int`**-Arithmetik

**`m = 1 + (month + 9) % 12`**

**`a = year - 1`** (für **`month < Mar`**), ansonsten **`year`**

**`y = a % 100`**

**`c = a / 100`**

Erweitern sie das Programm so, dass vor dem erfolgreichen Terminieren des Programms folgende Zeile (inklusive Zeilenumbruch) ausgegeben wird: ***`yyyy-mm-dd is a Ddd`***, wobei ***`yyyy`*** für das Jahr, ***`mm`*** für die Nummer des Monats (01…12) und ***`dd`*** für den Tag im Monat (01…). Z.B. **`2020-02-29 is a Sat`**.

**Vorgaben an die Umsetzung**

1. Definieren Sie einen **`enum`** Typen mit (**`typedef`**) Namen **`weekday_t`** dessen Werte die Englischen 3-Zeichen Abkürzungen der Tage sind, nämlich **`Sun`**, **`Mon`**, … **`Sat`** und stellen Sie sicher dass die Abkürzungen für die Werte **`0`**…**`6`** stehen.
2. Schreiben Sie eine Funktion **`weekday_t calculate_weekday(date_t date)`** nach der Beschreibung der obigen Formel. Das **`date`** Argument ist als gültig angenommen, d.h. es ist ein Programmier-Fehler, wenn das Programm diese Funktion mit einem ungültigen Datum aufruft. Machen Sie dafür als erste Codezeile in der Funktion eine Zusicherung (**`assert(is_valid_date(date));`**)
3. Schreiben Sie eine Funktion **`void print_weekday(weekday_t day)`**, welche für jeden gülteigen Tag eine Zeile auf **`stdout`** schreibt mit den Englischen 3-Zeichen Abkürzungen für den Wochentag, z.B. Sonntag: **`Sun`**, Montag: **`Mon`**, etc. Wenn ein ungültiger Wert für **`day`** erkannt wird, soll **`assert(!"day is out-of-range");`** aufgerufen werden.

**Hinweise**

- Für interessierte, siehe: https://de.wikipedia.org/wiki/Wochentagsberechnung

# 4 Aufgabe 2: Bounding-Box eines Dreiecks und Rechteck Vergleich

In dieser Aufgabe versucht das Programm ein Dreieck in ein rechteckiges Brett einzupassen. Dazu wird das Dreieck in Schritten von 1° gedreht: mit jeder Position wird das umschliessende Rechteck (Bounding Box) des Dreiecks berechnet und geprüft ob es in das Brett passt. Von den passenden Positionen wird die erste gewählt welche die kleinste Bounding Box Fläche hat.

Das Programm generiert schliesslich auf **`stdout`** Roh-Daten der geometrischen Figuren welche mit dem Bash Script **`tab2svg.sh`** in eine HTML/SVG Seite übersetzt werden kann.

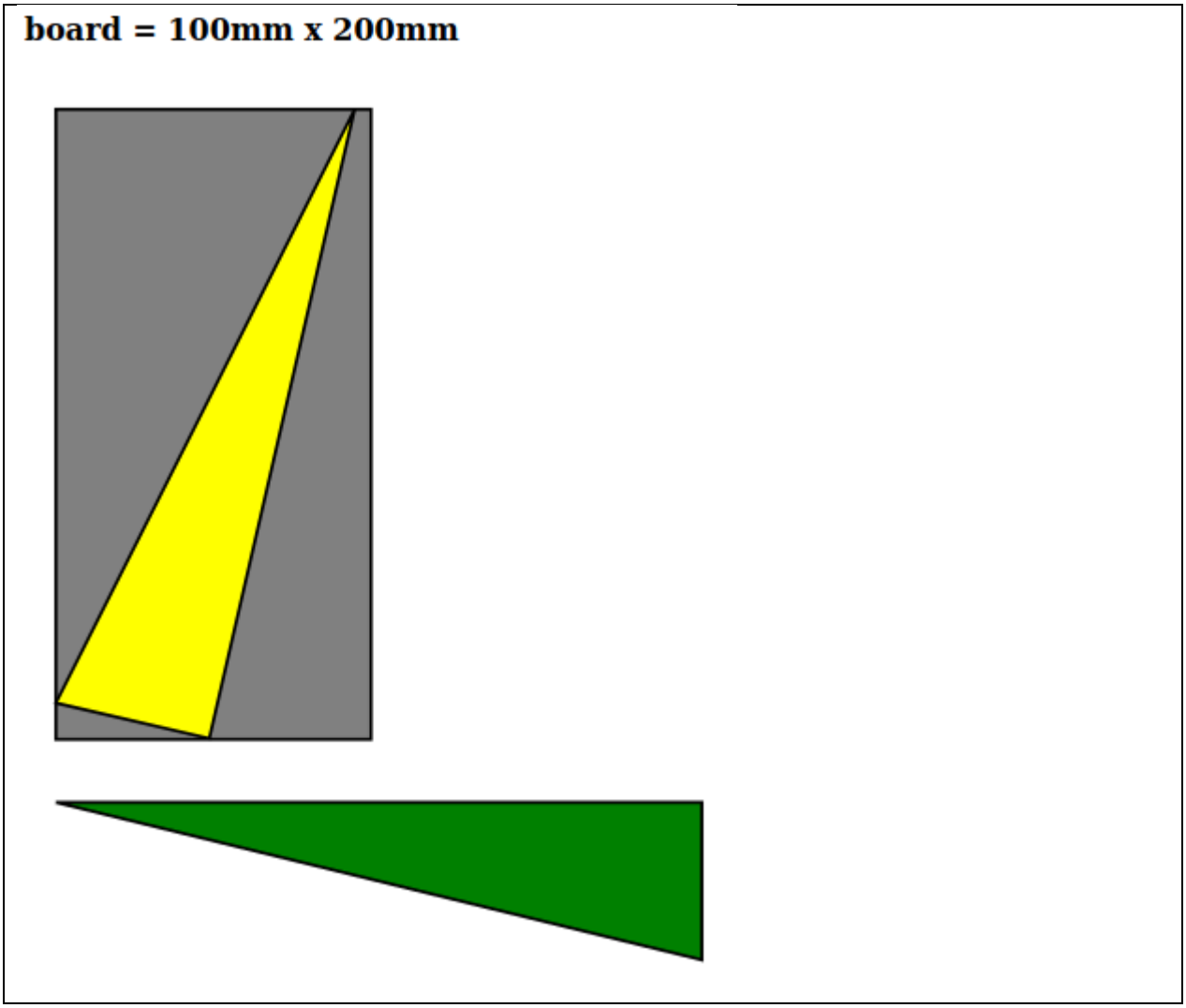
**Rohdaten (`bin/boundingbox 0/0-205/0-205/50 > raw.txt`)**

```
viewbox:-10.0:-10.0:225.0:290.0
rect:0.0:0.0:100.0:200.0:gray
polygon:0.0:220.0:205.0:220.0:205.0:270.0:green
polygon:94.8:0.0:48.7:199.7:0.0:188.5:yellow
```

**HTML/SVG Daten (`./tab2svg.sh < raw.txt > svg.html`)**

```
<!DOCTYPE html>
<html>
  <head>
    <title>SVG Data</title>
  </head>
  <body>
    <b>board = 100mm x 200mm</b>
    </p>
    <svg width="100mm" viewbox="-10.0 -10.0 225.0 290.0" xmlns="http://www.w3.org/2000/svg">
      <rect fill="gray" stroke="black" x="0.0" y="0.0" width="100.0" height="200.0"/>
      <polygon fill="green" stroke="black" points="0.0,220.0 205.0,220.0 205.0,270.0"/>
      <polygon fill="yellow" stroke="black" points="94.8,0.0 48.7,199.7 0.0,188.5"/>
    </svg>
  </body>
</html>
```

**Firefox Darstellung (`firefox svg.html`)**




Der Programm Rahmen kompiliert in der originalen Fassung nicht. Sie müssen beide Teilaufgaben lösen damit die Kompilation erfolgreich ist und die Tests ausgeführt werden können.

Prüfen Sie die Umsetzung beider Teilaufgaben mittels **`make test`**.

## 4.1 Teilaufgabe Fehlende Typen definieren

Im Projekt Rahmen fehlen folgende drei Typen welche Sie am gegebenen Ort im File main.c ergänzen sollen:

| Name | Art | Elemente |
|---|---|---|
| `point_t` | `struct` | `double x` und `double y` |
| `box_t` | `struct` | `point_t p`, `double w`, `double h` |
| `triangle_t` | `struct` | `point_t a`, `b` und `c` |

Solange diese Typen nicht oder falsch definiert sind, werden sie viele der folgenden oder ähnliche Fehlermeldungen sehen beim Aufruf von **`make`**:

```
error: unknown type name ‘point_t’
error: unknown type name ‘box_t’
error: unknown type name ‘triangle_t’
```

Nach Definition der obigen Typen fehlen nur noch vier Funktionen, welche in der folgenden Teilaufgabe hinzugefügt werden müssen.

### 4.2 Teilaufgabe Fehlende Funktionen implementieren

Nach der ersten Teilaufgabe fehlen folgende Funktionen welche Sie an den angegebenen Stellen im Code implementieren sollen.

- `static int compare_double(double a, double b)`
- `static int compare_area(box_t a, box_t b)`
- `static int is_zero_area(box_t box)`
- `static box_t triangle_bounding_box(triangle_t t)`

Konsultieren Sie die Beschreibung der Funktionen im Code.

## 5 Bewertung

Die gegebenenfalls gestellten Theorieaufgaben und der funktionierende Programmcode müssen der Praktikumsbetreuung gezeigt werden. Die Lösungen müssen mündlich erklärt werden.

| Aufgabe | Kriterium | Gewicht |
|---|---|---|
| 1 | Sie können das funktionierende Programm inklusive funktionierende Tests demonstrieren und erklären. | |
| | Teilaufgabe Argumente Parsen und auf Korrektheit prüfen | 1/4 |
| | Teilaufgabe Wochentag Berechnung | 1/4 |
| 2 | Sie können das funktionierende Programm inklusive funktionierende Tests demonstrieren und erklären. | |
| | Teilaufgabe Fehlende Typen definieren | 1/4 |
| | Teilaufgabe Fehlende Funktionen implementieren | 1/4 |

## 6 Anhang

### 6.1 Verwendete zusätzliche Sprach Elemente

| Sprach Element | Beschreibung |
|---|---|
| `int main(int argc, char *argv[]) {`<br>`  ...`<br>`}` | **`argc`**: Anzahl Einträge in **`argv`**.<br>**`argv`**: Array von Command Line Argumenten.<br>**`argv[0]`**: wie das Programm gestartet wurde<br>**`argv[1]`**: erstes Argument<br>…<br>**`argv[argc-1]`**: letztes Argument |

| Sprach Element | Beschreibung |
|---|---|
| `int a = 0;`<br>`int b = 0;`<br>`int c = 0;`<br>`int res = sscanf(argv[1]`<br>`, "%d-%d-%d"`<br>`, &a, &b, &c`<br>`);`<br>`if (res != 3) {`<br>`// Fehler Behandlung...`<br>`// ...`<br>`}` | Siehe **`man 3 sscanf`**.<br>Die Funktion **`sscanf`** gibt die Anzahl erfolgreich erkannte Argumente zurück. Unbedingt prüfen und angemessen darauf reagieren.<br>Die gelesenen Werte werden in **`a`**, **`b`** und **`c`**, gespeichert, dazu müssen Sie die Adresse der Variablen übergeben. Mehr Details dazu werden später erklärt. |
| `fprintf(stderr, "Usage: %s…\n", argv[0]);` | Siehe **`man 3 fprintf`**.<br>Schreibt formatierten Text auf den **`stderr`** Stream. |