Handout – Grundlagen

Editor

Tasks lösen mit dem Code::Blocks Template:

- Beschreibung und Beispiele vom Grader herunterladen.
- Code::Blocks mit unserem Template installieren.
- Neues Projekt mit Template erstellen.
- Lösung implementieren
- Testdaten im Ordner 'Others' hinzufügen.
- Programm mit F9 ausführen.

Einfaches Template

```
#include "soi.h" // stellt nützliche Features bereit
                 // nur mit dem SOI Template verfügbar

void main() {
  // dein Code
}
```

Befehle

Befehle enden immer mit einem Semikolon und werden von oben nach unten ausgeführt. Hier einige nützliche Befehle:

- Variable x vom Typ t deklarieren: `t x;`
- Du kannst mehrere Variablen aufs Mal deklarieren und ihnen Werte zuweisen: `int a = 10, b;`
- Neuer Wert einer bereit deklarierten Variable zuweisen: `a = 15;`
- Du kannst nicht den selben Namen für zwei Variablen verwenden (mehr dazu in späteren Vorträgen).

Einige Typen

- **`int`**: Ganzzahlen von $-2^{63} \approx -9 \cdot 10^{18}$ bis $2^{63} - 1 \approx 9 \cdot 10^{18}$.
- **`bool`**: Wahrheitswerte: `true` oder `false`.
- **`char`**: Einzelne Zeichen. Benutze einfache Anführungszeichen, zum Beispiel `char c = 'a';`
- `string`: Zeichenkette. Benutze doppelte Anführungszeichen, zum Beispiel `string s = "Hello, world";`

Ein- und Ausgabe

Lies die Eingabe mit `cin`:

```
int a, b;
cin >> a >> b;
```

Benutze `cout` um die Ausgabe auszugeben. `'\n'` erzeugt eine neue Zeile.

```
int a = 1, b = 2;
cout << "a+b: " << a+b << '\n'; // Gibt "a+b: 3" aus.
```

Operationen auf Ganzzahlen

Standard Operationen:

- `a+b`: Addition.
- `a-b`: Subtraktion.
- `a*b`: Multiplikation.
- `a/b`: Ganzzahldivision (z. B. `5/3=1`).
- `a%b`: Rest einer Ganzzahldivision von a und b (z. B. `5%3=2`). Auch 'Modulo' genannt.

Gib Acht auf Operatorenpräzedenz. Multiplikationen, Divisionen und Modulos werden vor Additionen und Subtraktionen ausgeführt.
Operationen mit der gleichen Präzedenz werden von links nach rechts ausgeführt.
Nutze Klammern um eine andere Ordnung zu erzwingen: `(a+b)*c-d/(e%(f*a))`.

Kommentare

Mit Kommentaren kannst du den Code einfacher lesbar machen (auch wenn du der Einzige bist, der den Code liest). Kommentare beginnen mit `//` und werden vom Compiler ignoriert:

```
cout << (h2-h1)/(x2-x1) << '\n'; // Steigung ausgeben
```

Bedingungen

Du kannst bedingte Ausdrücke verwenden, damit das Programm je nach Wahrheitswert eines Ausdruckes etwas anderes tut.
Benütze hierfür ein **`if`**.
Erstelle einen Codeblock zwischen Klammern `{ ... }` um die Befehle anzugeben, die nur ausgeführt werden, falls die Bedingung erfüllt ist.

```
if (condition) {
    // nur ausgeführt, falls condition wahr ist
    ...
}
```

Du kannst **`else`** verwenden, um Code anzugeben, der ausgeführt wird, falls die Bedingung nicht erfüllt ist:

```
if (condition) {
    // nur ausgeführt, falls condition wahr ist
    ...
} else {
    // nur ausgeführt, falls condition falsch ist
    ...
}
```

Bedingungen sollten Ausdrücke sein, die ein Wahrheitswert (`true` oder `false`) zurück geben. Hier einige nützliche Ausdrücke, um einige Bedingungen mit zwei Zahlen a und b zu testen:

- `a==b` ist `true` falls a und b gleich sind und `false` sonst.
- `a!=b` ist `true` falls a und b nicht gleich sind und `false` sonst.
- `a<b` ist `true` falls a kleiner ist als b und `false` sonst.
- `a>b` ist `true` falls a grösser ist als b und `false` sonst.
- `a<=b` ist `true` falls a kleiner oder gleich ist als b und `false` sonst.
- `a>=b` ist `true` falls a grösser oder gleich ist als b und `false` sonst.

Du kannst mehrere Wahrheitswerte mit logischen Operatoren kombinieren:

- `!a` ist `true` falls a `false` ist und `false` sonst.
- `a&&b` ist `true` falls sowohl a als auch b `true` sind und `false` sonst.
- `a||b` ist `true` falls entweder a oder b `true` ist und ist `false` falls beide `false` sind.

Die Operatorenpräzedenz ist: `!` > numerische Operatoren > `==`, `!=` > `||` > `&&`.
Du kannst Bedingungen auch verschachteln:

```
if (x == 2) {
    ...
    if (a - 3 != 5) {
        ...
    }
    ...
}
```

Weiteres

Um mehr über C++ zu lernen, höre dir die folgenden Vorträge an, frag uns und zögere nicht, auf Referenzseiten wie `https://en.cppreference.com/w/` nachzuschauen.