Übungsstunde

Woche 10

Adel Gavranović
adel.gavranovic@inf.ethz.ch

Overview

Heutige Themen

Intro

Self-Assessment III

Objekte

Overloading

Outro

Links

- polybox zum Material für die Übungsstunden
- Mail an Assistenten

Follow-up aus vorheriger Übungsstunde

Follow-up aus vorheriger Übungsstunde

- Vielen dank für das Feedback!

Kommentare zu [code]expert

Allgemein

Kommentare zu [code]expert

Allgemein

- Anzahl Abgaben sind runter. Bitte versucht dranzubleiben! Die nächsten Themen sind unabhängig von den jetzigen, also Einsteigen ist recht einfach...

Kommentare zu [code]expert

Allgemein

- Anzahl Abgaben sind runter. Bitte versucht dranzubleiben! Die nächsten Themen sind unabhängig von den jetzigen, also Einsteigen ist recht einfach...
- **Empfehlung**: auch unvollendete Aufgaben können abgegeben werden (hilft euch, da ich die Session dann in die Richtung lenken kann und auf Missverständnisse eingehen kann). Dort ist aber wichtig, dass ihr mir `// als Kommentar` mitteilt was genau nicht ging

Kommentare zu [code]expert

Allgemein

- Anzahl Abgaben sind runter. Bitte versucht dranzubleiben! Die nächsten Themen sind unabhängig von den jetzigen, also Einsteigen ist recht einfach...
- **Empfehlung**: auch unvollendete Aufgaben können abgegeben werden (hilft euch, da ich die Session dann in die Richtung lenken kann und auf Missverständnisse eingehen kann). Dort ist aber wichtig, dass ihr mir `// als Kommentar` mitteilt was genau nicht ging

E8:T1 ”Vector and Matrix Operations”

- Idealerweise ist eure `main()` einfach nur noch ein Funktionsaufruf nach dem anderen

Kommentare zu [code]expert

Allgemein

- Anzahl Abgaben sind runter. Bitte versucht dranzubleiben! Die nächsten Themen sind unabhängig von den jetzigen, also Einsteigen ist recht einfach...
- **Empfehlung**: auch unvollendete Aufgaben können abgegeben werden (hilft euch, da ich die Session dann in die Richtung lenken kann und auf Missverständnisse eingehen kann). Dort ist aber wichtig, dass ihr mir `// als Kommentar` mitteilt was genau nicht ging

E8:T1 ”Vector and Matrix Operations”

- Idealerweise ist eure `main()` einfach nur noch ein Funktionsaufruf nach dem anderen
- Versucht Initialization erst im allerletzten Moment zu machen. Gibt es einen Grund, wieso das die meisten nicht gemacht haben?

Kommentare zu [code]expert

E8:T2 ”Decode Binary NZZ Front Page”

Kommentare zu [code]expert

E8:T2 ”Decode Binary NZZ Front Page”

- `const std::string&`

Kommentare zu [code]expert

E8:T2 ”Decode Binary NZZ Front Page”

- `const std::string&`
- führt keine Variablen ein, wenn es nicht nötig ist (z.B. wenn man eine Zahl nur einmal verwendet, braucht man dafür wirklich keinen Namen)

Kommentare zu [code]expert

E8:T2 ”Decode Binary NZZ Front Page”

- `const std::string&`
- führt keine Variablen ein, wenn es nicht nötig ist (z.B. wenn man eine Zahl nur einmal verwendet, braucht man dafür wirklich keinen Namen)

E8:T3 ”Recursive Function Analysis”

Kommentare zu [code]expert

E8:T2 ”Decode Binary NZZ Front Page”

- `const std::string&`
- führt keine Variablen ein, wenn es nicht nötig ist (z.B. wenn man eine Zahl nur einmal verwendet, braucht man dafür wirklich keinen Namen)

E8:T3 ”Recursive Function Analysis”

- seid spezifischer. nicht *”es” sondern* n *ist streng monoton fallend, weil . . . .*. Es wäre doof, deswegen Punkte liegen zu lassen

Kommentare zu [code]expert

E8:T2 ”Decode Binary NZZ Front Page”

- `const std::string&`
- führt keine Variablen ein, wenn es nicht nötig ist (z.B. wenn man eine Zahl nur einmal verwendet, braucht man dafür wirklich keinen Namen)

E8:T3 ”Recursive Function Analysis”

- seid spezifischer. nicht *”es” sondern* n *ist streng monoton fallend, weil . . . .*. Es wäre doof, deswegen Punkte liegen zu lassen
- vergesst nicht den Base Case (meist `n == 0`) zu nennen

Kommentare zu [code]expert

E8:T2 ”Decode Binary NZZ Front Page”

- `const std::string&`
- führt keine Variablen ein, wenn es nicht nötig ist (z.B. wenn man eine Zahl nur einmal verwendet, braucht man dafür wirklich keinen Namen)

E8:T3 ”Recursive Function Analysis”

- seid spezifischer. nicht *”es” sondern* n *ist streng monoton fallend, weil . . . .*. Es wäre doof, deswegen Punkte liegen zu lassen
- vergesst nicht den Base Case (meist `n == 0`) zu nennen
- benutzt ruhig mathematische Notation in den PRE/POSTs. Das ist meistens klarer als Worte

Fragen zu [code]expert eurerseits?

Lernziele

- ☐ `class` und `struct` definieren und mit ihnen umgehen können
- ☐ Operatoren *overloaden* können

Self-Assessment III

Self-Assessment III

- ☐ Als Hausaufgabe (via moodle)

`struct` **vs** `class`

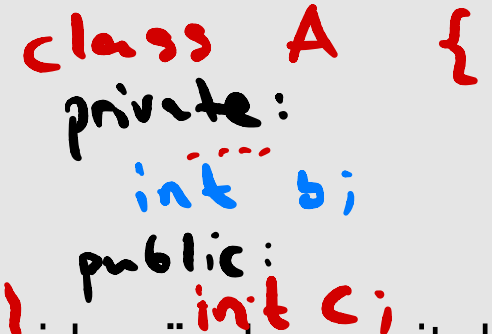
Unterschied?

`struct` vs `class`

Unterschied?

Der einzige Unterschied liegt in der default *visibility*.

Object	Default Visibility
`struct`	`public` (”visible”)
`class`	`private` (”invisible”)



Man kann die visibility der Members bei beiden ändern, mit den keywords `private:` bzw. `public:`.

`struct` vs `class`

Unterschied?

Der einzige Unterschied liegt in der default *visibility*.

Object	Default Visibility
`struct`	`public` (”visible”)
`class`	`private` (”invisible”)

Man kann die visibility der Members bei beiden ändern, mit den keywords `private:` bzw. `public:` .

Wann sollte man was verwenden?

`struct` vs `class`

Unterschied?

Der einzige Unterschied liegt in der default *visibility*.

Object	Default Visibility
`struct`	`public` (”visible”)
`class`	`private` (”invisible”)

Man kann die visibility der Members bei beiden ändern, mit den keywords `private:` bzw. `public:` .

Wann sollte man was verwenden?

Eigentlich egal; Hauptsache die visibility stimmt.
Empfehlung: `class` für komplizierteres Zeugs das hauptsächtlich viele Memberfunktionen hat und `struct` für Datenbündel (Punkte, Personen, mathematische Objekte)

Fragen/Unklarheiten?

Function Overloading

”Wie weiss der Computer, welche Funktion man aufrufen möchte, wenn mehrere Funktionen denselben Namen haben?”

Function Overloading

”Wie weiss der Computer, welche Funktion man aufrufen möchte, wenn mehrere Funktionen denselben Namen haben?”
$\rightarrow$ Woran kann der Compiler gleichnamige Funktionen unterscheiden?

Function Overloading

”Wie weiss der Computer, welche Funktion man aufrufen möchte, wenn mehrere Funktionen denselben Namen haben?”
$\rightarrow$ Woran kann der Compiler gleichnamige Funktionen unterscheiden?

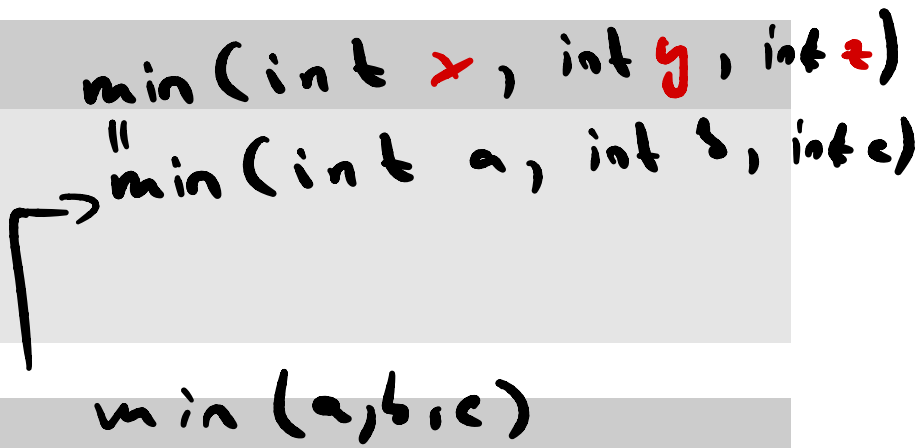
Kriterien

Function Overloading

”Wie weiss der Computer, welche Funktion man aufrufen möchte, wenn mehrere Funktionen denselben Namen haben?”
→ Woran kann der Compiler gleichnamige Funktionen unterscheiden?

Kriterien

- Anzahl der Funktionsparameter
- Typ der Funktionsparameter



keine Kriterien

Function Overloading

”Wie weiss der Computer, welche Funktion man aufrufen möchte, wenn mehrere Funktionen denselben Namen haben?”
$\rightarrow$ Woran kann der Compiler gleichnamige Funktionen unterscheiden?

Kriterien

- Anzahl der Funktionsparameter
- Typ der Funktionsparameter

keine Kriterien

- Name der Funktionsparameter
- Return-Typ der Funktion

Function Overloading

Was juckt uns das, dass zwei Funktionen den gleichen Namen haben können?

[1] ▸ cppreference

Function Overloading

Was juckt uns das, dass zwei Funktionen den gleichen Namen haben können?
$\rightarrow$ Weil wir so *Operator Overloading* viel besser verwenden können!

[1] ▸ cppreference

Function Overloading

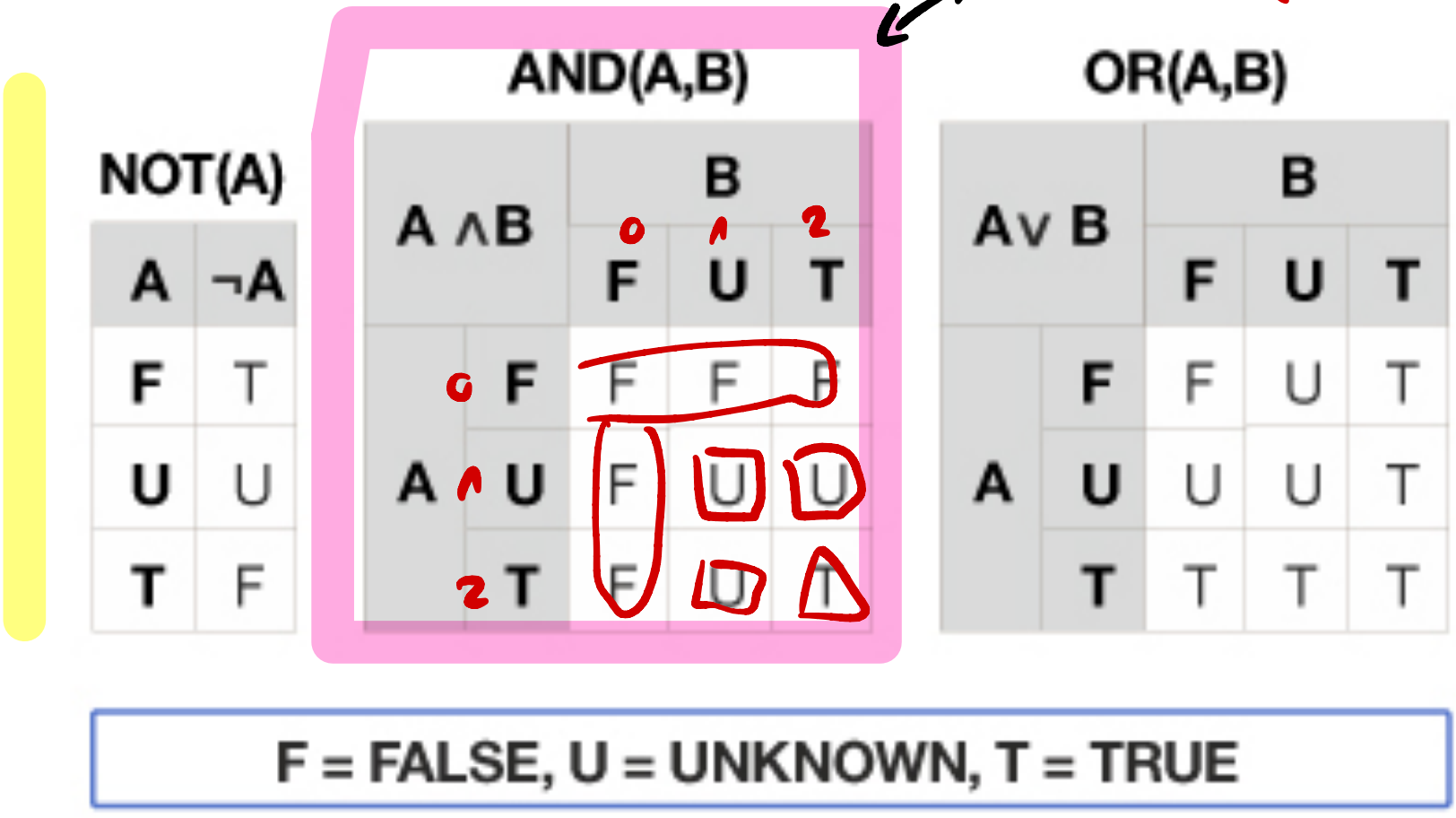
Was juckt uns das, dass zwei Funktionen den gleichen Namen haben können?
$\rightarrow$ Weil wir so *Operator Overloading* viel besser verwenden können!

Operatoren (`*`, `+`, `=`, uvm.[1]) sind eigentlich nur fancy Funktionen. Der Operator `*` beispielsweise hat mehrere Bedeutungen. Die richtige Deutung erhält man durch die Inputs. So kann man eine Art Multiplikation mit `*` einführen, die für `std::vector<double>` etwas anderes macht als für `int`.

[1] ▸ cppreference

Aufgabe "Tribool"

Tribool: ein `bool` , aber mit drei möglichen Werten: `false` , `unknown` , und `true` .



NOT(A)

A	¬A
F	T
U	U
T	F

AND(A,B)

A ∧ B	B = F	B = U	B = T
A = F	F	F	F
A = U	F	U	U
A = T	F	U	T

OR(A,B)

A ∨ B	B = F	B = U	B = T
A = F	F	U	T
A = U	U	U	T
A = T	T	T	T

F = FALSE, U = UNKNOWN, T = TRUE

Aufgabe ”Tribool”

Diese Implememtation speichert die Wahrheitswerte in einer `class` namens `Tribool` und zwar als `private unsigned int`.

Aufgabe ”Tribool”

Diese Implememtation speichert die Wahrheitswerte in einer `class` namens `Tribool` und zwar als `private unsigned int`.

- Wieso ist das eine gute Idee?
- Wie könnte man den Wahrheitswert speichern?

Aufgabe ”Tribool”

Diese Implememtation speichert die Wahrheitswerte in einer `class` namens `Tribool` und zwar als `private unsigned int`.

- Wieso ist das eine gute Idee?
- Wie könnte man den Wahrheitswert speichern?
- Diese Aufgabe gehen wir zusammen an

Aufgabe ”Tribool” Konzepte

Folgenden Konzepten und Keywords werden wir beim Lösen dieser Aufgabe begegnen:

- Classes und Structs
- Visibility
- Operator Overloading
- Deklaration vs Definition
- Out-of-Class-Definitionen
- `const` -Funktionen
- Konstruktoren (”C-tors”)
- Member Initializer Lists
- ...

Aufgabe ”Tribool”

- **Step 1:** Den ersten Konstruktor implementieren wir zusammen. Den zweiten versucht ihr selbst zu implementieren

Aufgabe ”Tribool”

- **Step 1:** Den ersten Konstruktor implementieren wir zusammen. Den zweiten versucht ihr selbst zu implementieren
- **Step 2:** Versucht Step 2 ebenfalls selbst

Aufgabe ”Tribool”

- **Step 1:** Den ersten Konstruktor implementieren wir zusammen. Den zweiten versucht ihr selbst zu implementieren
- **Step 2:** Versucht Step 2 ebenfalls selbst
- **Step 3:** Zuerst kurze Diskussion zu einer kleveren Implementationsmöglichkeit und dann könnt ihr es selbst versuchen.

Bis zum nächsten Mal

- ☐ Self-Assessment III
- ☐ macht eure Hausaufgaben
- ☐ bleibt gesund

Allgemeine Fragen?

"INV": "Invariant" ← soll "immer*" wahr sein

(a ∈ {0,1,2}),
(b <= 5), etc...

*falls die "Invariante" vor Funktionsaufruf "true" war, so muss sie auch nach dem F.-aufruf "true" sein. (zwischenzeitlich kann die INV kurz false sein)