ETH zürich

Übungsstunde W02

Computer Science (CSE) – AS 23

Übersicht

Heutiges Programm

Kennenlernen
Organisatorisches
Integer Division & Modulo
Binärrepräsentation
Expressions und Evaluationen
Tipps zu `[code]`expert
Outro

`rwko.ch/lily`

Intro

Herzlich Willkommen!

Ziele

Ziele für heute

- ☐ Uns kennenlernen
- ☐ Organisatorisches besprechen
- ☐ `[code]`expert einrichten
- ☐ Unklarheiten klären
- ☐ Int-Division, Operatoren und Modulo verstehen
- ☐ Zahlen in andere Basen umrechnen können
- ☐ Valide C++-Expressions erkennen
- ☐ C++-Expressions evaluieren können

1. Kennenlernen

Eure Assistenz

Lily

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren

Eure Assistenz

Lily

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren
- Hat Informatik gut bestanden und möchte euch auch durchbringen

Eure Assistenz

Lily

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren
- Hat Informatik gut bestanden und möchte euch auch durchbringen
- Hier, um euch zu helfen

Eure Assistenz

Lily

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren
- Hat Informatik gut bestanden und möchte euch auch durchbringen
- Hier, um euch zu helfen
- Tanzt gerne Ballett und Modern
- Aus Zürich

Eure Kommiliton:innen

Stell dich vor!

Eure Kommiliton:innen

Stell dich vor!

- Wie heisst du?
- Woher kommst du?
- Wieso RW an der ETH?
- Hast du Erfahrung im Programmieren (insb. C++)?
- Hast du coole Hobbies oder aussergewöhnliche Interessen?

Fragen/Unklarheiten?

2. Organisatorisches

Informatik

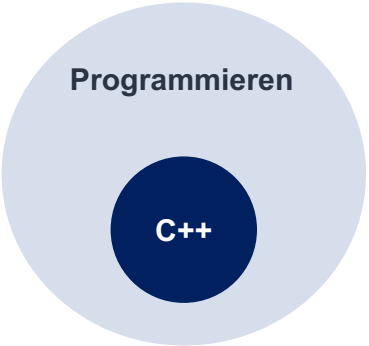
Informationen zur Organisation

Kennenlernen

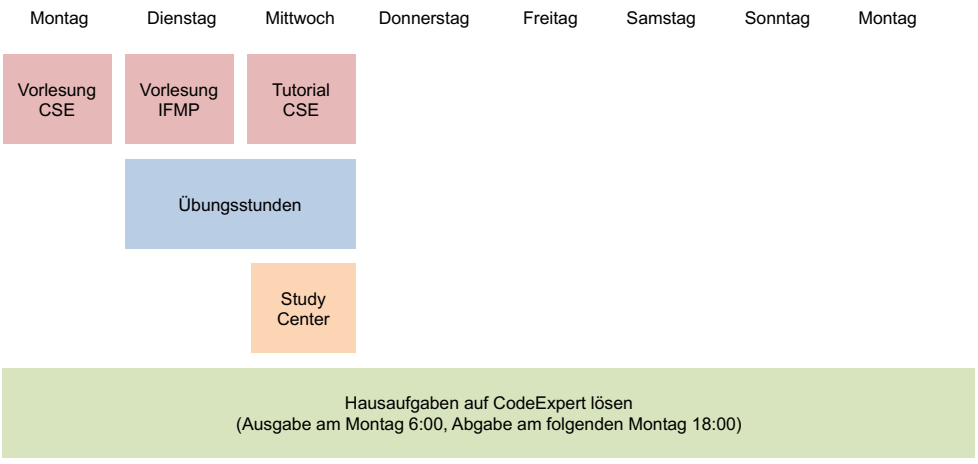
- Name?
- Welcher Teilbereich deines Studiums interessiert dich am meisten?

Ziel des Kurses

- Vorlesung
- Übungsstunde
- Wöchentliche Übungen
- Bonusaufgaben
- Study Center



Wöchentlicher Plan



Wöchentliche Übungen und Bonusaufgaben

- Alle Übungen findet ihr auf [code]expert (https://expert.ethz.ch).
- Ihr müsst euch zuerst mithilfe dem euch gesandten Link in die Übungsgruppe einschreiben.

- **Wöchentliche Übungen:**
  - Zweck: Neuen Stoff anwenden.
  - Veröffentlichung: Montag um 6:00.
  - Deadline: Eine Woche später (Montag 18:00).
  - Erlaubt euch Erfahrungspunkte (XP) zu sammeln.

- **Bonusaufgaben** (benötigt ca. 2/3 der Erfahrungspunkte zur Freischaltung):
  - Zweck: Wissen zu verschiedenen Themen kombinieren.
  - Erlaubt euch max. +0.25 Bonus für die Endnote zu erreichen (mit 2/3 der Bonuspunkte).

Wöchentliche Übungen und Bonusaufgaben

- Alle Übungen findet ihr auf [code]expert (https://expert.ethz.ch).
- Ihr müsst euc[illegible]ungsgruppe einschreiben
- **Wöchentlich[illegible]**
  - Zweck: Neu[illegible]
  - Veröffentlic[illegible]
  - Deadline: E[illegible]
  - Erlaubt euc[illegible]
- **Bonusaufga[illegible]**chaltung):
  - Zweck: Wissen zu verschiedenen Themen kombinieren.
  - Erlaubt euch max. +0.25 Bonus für die Endnote zu erreichen (mit 2/3 der Bonuspunkte).

Wichtig!

Benutzt zum Lösen der Übungsaufgaben **nur** das, was bereits in der Vorlesung vorgestellt wurde und was nicht von der Aufgabenstellung verboten wird. Beachtet, dass der Autograder weitere Einschränkungen (wie die Vermeidung von globalen Variablen) auferlegen kann, die nicht ausdrücklich in der Aufgabenstellung angegeben sind. Beachtet außerdem, dass Warnungen als Fehler behandelt werden. Überprüft also die Ausgabe des Autograders sorgfältig, um zu vermeiden, dass ihr 0 Punkte bekommt.

Jede Woche wird eine Zusammenfassung der in dieser Woche vorgestellten Konzepte herausgegeben.

Übungsstunden

- Zweck: Vorbereitung für das Lösen von zukünftigen und vergangenen Übungen.
- Ansatz: Vor allem interaktiver Unterricht und konstruktive Diskussionen.
- Wir erwarten, dass ihr:
  - Aktiv am Unterricht teilnehmt
  - Fragen stellt, wenn ihr den Inhalt nicht versteht, oder warum wir ein bestimmtes Thema behandeln
- Es ist völlig normal, wenn ihr Fehler macht, wir sind im Lernprozess. Bitte vermeidet es, Sachen zu tun, welche andere stören könnten. Wenn eine Aufgabe zu einfach ist, helft anderen.

Study Center

- Zweck: Eine Möglichkeit, um nach individueller Hilfe zum Kurs zu fragen

IFMP:

- Zeit: Mittwoch 16:15-18:15, beginnend ab dem 27. September
- Ort: Mensa Polyterrasse
- Link: https://studycenter.ethz.ch/

CSE (mit MAVT geteilt):

- Zeit: Mittwoch 18:15-20:00, beginnend ab dem 4. Oktober
- Ort: HG F3

Informationen & Kontakt

- Mehr Informationen sind auf dem Informationsblatt zur Organisation zu finden.
- Für Fragen betreffend der *Vorlesungsinhalte* könnt ihr während der Vorlesung fragen.
- Für Fragen betreffend der *Übungen* könnt ihr euren TA fragen.
- Für *administrative* Fragen kontaktiert bitte den Head-TA (siehe Website für die E-Mail-Adresse).

Fragen/Unklarheiten?

Zur Übungsstunde

- Slides werde ich auf Notion hochladen (wenn möglich vor der Stunde und eine korrigierte Version danach)

Zur Übungsstunde

- Slides werde ich auf Notion hochladen (wenn möglich vor der Stunde und eine korrigierte Version danach)
- Wird *nicht* aufgezeichnet

Zur Übungsstunde

- Slides werde ich auf Notion hochladen (wenn möglich vor der Stunde und eine korrigierte Version danach)
- Wird *nicht* aufgezeichnet
- Bitte stellt Fragen bei Unklarheiten
- Bitte beteiligt euch am Unterricht
- Bitte korrigiert mich, sobald ich Fehler mache

Zur Übungsstunde

- Slides werde ich auf Notion hochladen (wenn möglich vor der Stunde und eine korrigierte Version danach)
- Wird *nicht* aufgezeichnet
- Bitte stellt Fragen bei Unklarheiten
- Bitte beteiligt euch am Unterricht
- Bitte korrigiert mich, sobald ich Fehler mache
- Ihr könnt Fragen auch direkt in euren Code bei `[code]`expert schreiben, aber das sehe ich erst *nach* der Abgabe

```
int a = 42; // Eure Kommentare und Fragen hierhin
```

- Fragen via Mail sind immer willkommen

How to [code]expert

(öffnet **expert.ethz.ch** auf dem Gerät eurer Wahl)

- [code]expert kann ein wenig pingelig sein

How to [code]expert

(öffnet **expert.ethz.ch** auf dem Gerät eurer Wahl)

- [code]expert kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)

How to [code]expert

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- [code]expert kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.

How to [code]expert

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- [code]expert kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.
- Textaufgaben werden vollständig manuell korrigiert

How to [code]expert

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- [code]expert kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.
- Textaufgaben werden vollständig manuell korrigiert
- Feedback zu den Aufgaben könnt ihr innerhalb einer Woche (von mir) erwarten (falls dringend, einfach eine Mail an mich)

Fragen/Unklarheiten?

3. Integer Division & Modulo

Der Unterschied zwischen = und ==

Assignment Operator (=)

gebraucht, um Variablen Werte zuzuweisen

Der Unterschied zwischen = und ==

Assignment Operator (=)

gebraucht, um Variablen Werte zuzuweisen

```
int a = 42; // assignes the value 42 to the variable a
int b = 18; // assignes the value 18 to the variable b
```

Equality Operator (==)

gebraucht, um Gleichheit zwischen Variablen zu überprüfen

Der Unterschied zwischen = und ==

Assignment Operator (=)

gebraucht, um Variablen Werte zuzuweisen

```
int a = 42; // assignes the value 42 to the variable a
int b = 18; // assignes the value 18 to the variable b
```

Equality Operator (==)

gebraucht, um Gleichheit zwischen Variablen zu überprüfen

```
(a == b) // this "expression" will equal 1 (true)
         // or 0 (false) ("boolean")
```

Integer Division & Modulo

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler “ignoriert” Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler “ignoriert” Dezimalstellen, wenn er (`unsigned`) `int` durch (`unsigned`) `int` teilt.

Modulo (`a%b`)

Gibt den Rest einer Division an.

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler “ignoriert” Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 ==
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler “ignoriert” Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 == 2
15/4 ==
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 == 2
15/4 == 3
16/4 ==
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 == 2
15/4 == 3
16/4 == 4
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

```
7%3 ==
```

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 == 2
15/4 == 3
16/4 == 4
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

```
7%3 == 1
15%4 ==
```

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 == 2
15/4 == 3
16/4 == 4
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

```
7%3 == 1
15%4 == 3
16%4 ==
```

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 == 2
15/4 == 3
16/4 == 4
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

```
7%3 == 1
15%4 == 3
16%4 == 0
```

Integer Division & Modulo

Integer Division (`a/b`)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**`unsigned`**) **`int`** durch (**`unsigned`**) **`int`** teilt.

```
7/3 == 2
15/4 == 3
16/4 == 4
```

Modulo (`a%b`)

Gibt den Rest einer Division an.

```
7%3 == 1
15%4 == 3
16%4 == 0
```

wichtige Identität

```
(a / b) * b + a % b == a
```

Fragen/Unklarheiten?

Mal schauen, was ihr gelernt habt

- Geht auf `expert.ethz.ch`
- Logt euch ein
- Geht zu “Code Examples”
- Unter “Lecture 2: Exercise Session”, öffnet “Last Three Digits”

Mal schauen, was ihr gelernt habt

- Geht auf `expert.ethz.ch`
- Logt euch ein
- Geht zu “Code Examples”
- Unter “Lecture 2: Exercise Session”, öffnet “Last Three Digits”
- Versucht die Aufgabe zu lösen (10 Minuten)
- Wir schauen uns eure Ansätze später an

Aufgabe

Task “Last Three Digits”

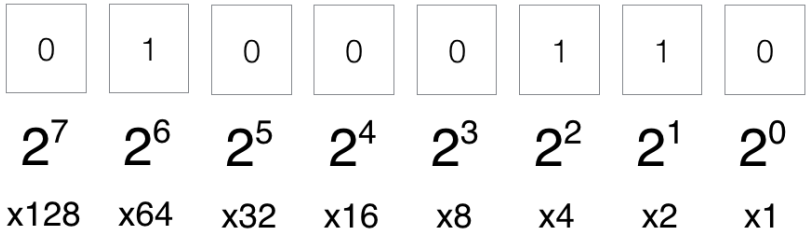
Write a program which reads in an integer **a** larger than 1000 and outputs its last three digits with a space between them.

For example, if **a** = 14325, the output should be 3 2 5.

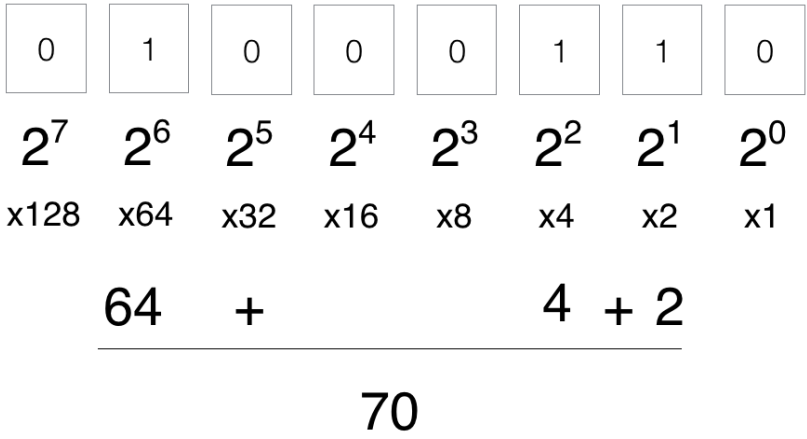
4. Binärrepräsentation

Binärrepräsentation

Binärrepräsentation



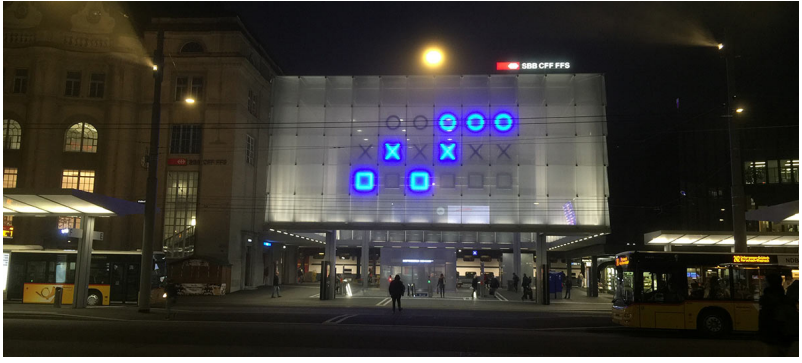
Binärrepräsentation



Bahnhofsuhr

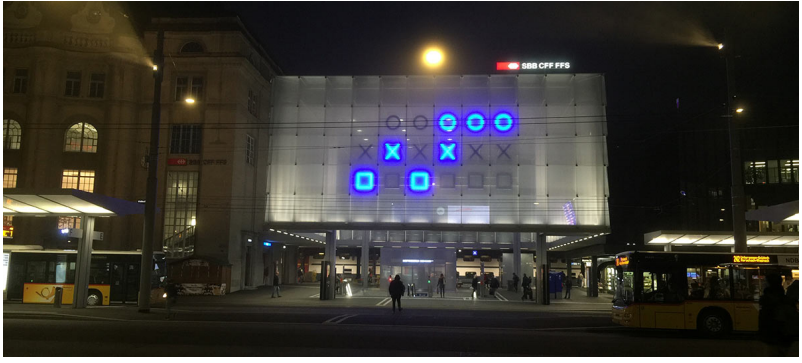
Bahnhofsuhr

Wie spät ist es?



Bahnhofsuhr

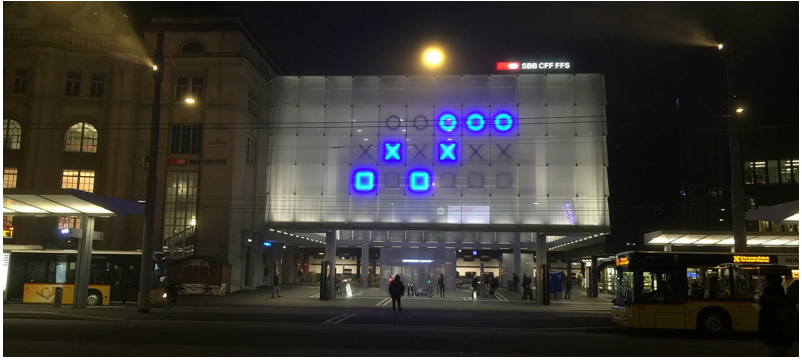
Wie spät ist es?



Source: `bahnhofsuhrsg.ch`

Bahnhofsuhr

Wie spät ist es?



Source: `bahnhofsuhrsg.ch`

07 : 20 : 40

Binärrepräsentation

Aufgabe

Wie sieht ein Algorithmus aus, der die Binärrepräsentation einer Dezimalzahl herausfindet?

Wie würde man beispielsweise bei der Dezimalzahl 61_{10} vorgehen, um sie in Binärrepräsentation zu bringen?

Tipp: Überlegt euch, wie wir bei three last digits (also im Dezimalsystem) vorgegangen sind

Binärrepräsentation

Aufgabe

Wie sieht ein Algorithmus aus, der die Binärrepräsentation einer Dezimalzahl herausfindet?

Wie würde man beispielsweise bei der Dezimalzahl 61_{10} vorgehen, um sie in Binärrepräsentation zu bringen?
Tipp: Überlegt euch, wie wir bei three last digits (also im Dezimalsystem) vorgegangen sind

Lösung

Teile die Dezimaldarstellung durch 2 und behalte den Rest (wie eine Modulodivision). Teile dann die übrige Nummer nochmal und so weiter, bis man 0 erreicht.

Binärrepräsentation

$$
\begin{aligned}
61 &= 2 * 30 + 1 \\
30 &= 2 * 15 + 0 \\
15 &= 2 * 7 + 1 \\
7 &= 2 * 3 + 1 \\
3 &= 2 * 1 + 1 \\
1 &= 2 * 0 + 1
\end{aligned}
$$

Binärrepräsentation

$$
\begin{aligned}
61 &= 2 * 30 + 1 \\
30 &= 2 * 15 + 0 \\
15 &= 2 * 7 + 1 \\
7 &= 2 * 3 + 1 \\
3 &= 2 * 1 + 1 \\
1 &= 2 * 0 + 1
\end{aligned}
$$

Dann die letzte Spalte von unten nach oben ablesen und fertig!

$$61_{10} = 111101_2$$

Binärrepräsentation von negativen Integern

Aufgabe

Finde einen Weg, um negative Integer abzuspeichern.

Tipp:

$$n + (-n) = 0$$

Binärrepräsentation von negativen Integern

Aufgabe

Finde einen Weg, um negative Integer abzuspeichern.

Tipp:

$$n + (-n) = 0$$

Lösung

Behandle die vorderste Ziffer als das negative ihres “eigentlichen” Werts

▸ Sehr gutes Video dazu

Binärrepräsentation von negativen Integern

Aufgabe

Wie erhält man die (*signed*) `int`-Repräsentation einer beliebigen, negativen Ganzzahl $n < 0$?

Binärrepräsentation von negativen Integern

Aufgabe

Wie erhält man die (*signed*) `int`-Repräsentation einer beliebigen, negativen Ganzzahl $n < 0$?

Lösung

1. Absolutbetrag von n bestimmen
2. Absolutbetrag von n in Binärrepräsentation aufschreiben
3. Bits flippen
4. 1 addieren

Fragen/Unklarheiten?

5. Expressions und Evaluationen

Was sind Expressions?

Expressions

…sind Ausdrücke die evaluiert werden können. Sie kommen oft in Form von mathematischen Ausdrücken vor, die man dann Stück für Stück evaluiert.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

[1] Hier steht `u` für `unsigned int`. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

```
5u + 5 * 3u
```

[1] Hier steht `u` für **`unsigned int`**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

`5u + 5 * 3u` Punkt vor Strich

[1] Hier steht `u` für `unsigned int`. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

`5u + 5 * 3u` Punkt vor Strich

`5u + (5 * 3u)`

[1] Hier steht `u` für **`unsigned int`**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

`5u + 5 * 3u` Punkt vor Strich

`5u + (5 * 3u)` das Resultat wird zu **`unsigned int`**

[1] Hier steht `u` für **`unsigned int`**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

`5u + 5 * 3u` Punkt vor Strich

`5u + (5 * 3u)` das Resultat wird zu **`unsigned int`**

`5u + 15u`

[1] Hier steht `u` für **`unsigned int`**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

`5u + 5 * 3u` Punkt vor Strich

`5u + (5 * 3u)` das Resultat wird zu **`unsigned int`**

`5u + 15u` simple Addition

[1] Hier steht `u` für **`unsigned int`**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression[1] `5u + 5 * 3u`

`5u + 5 * 3u` Punkt vor Strich

`5u + (5 * 3u)` das Resultat wird zu **`unsigned int`**

`5u + 15u` simple Addition

`20u`

[1] Hier steht `u` für **`unsigned int`**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Valide Expression erkennen

Welche der folgenden Snippets ist eine valide C++-Expression und welche nicht?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

[2]Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets ist eine valide C++-Expression und welche nicht?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

[2]Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets ist eine valide C++-Expression und welche nicht?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. ist nicht valide, da die Klammer `(` nicht geschlossen wird

[2] Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets ist eine valide C++-Expression und welche nicht?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. ist nicht valide, da die Klammer (`(`) nicht geschlossen wird
- 4. ist nicht valide, da `(a*3)` zu einem R-value wird, der Assignment-Operator (`=`) aber ein L-value[2] zu seiner Linken erwartet

[2] Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets ist eine valide C++-Expression und welche nicht?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. ist nicht valide, da die Klammer (`(`) nicht geschlossen wird
- 4. ist nicht valide, da `(a*3)` zu einem R-value wird, der Assignment-Operator (`=`) aber ein L-value[2] zu seiner Linken erwartet
- 1. und 2. sind valide C++-Expressions

[2]Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets sind L-Values und welche sind R-Values?[3]

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

[3] Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets sind L-Values und welche sind R-Values?[3]

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

[3] Wurde in Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets sind L-Values und welche sind R-Values?[3]

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. und 4. sind weder noch, da sie keine validen Expressions sind

[3] Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets sind L-Values und welche sind R-Values?[3]

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. und 4. sind weder noch, da sie keine validen Expressions sind
- 1. ist ein R-Value, da der Multiplikation-Operator (`*`) ein R-Value zurückgibt

[3] Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Welche der folgenden Snippets sind L-Values und welche sind R-Values?[3]

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. und 4. sind weder noch, da sie keine validen Expressions sind
- 1. ist ein R-Value, da der Multiplikation-Operator (`*`) ein R-Value zurückgibt
- 2. ist ein L-Value, da der Assignment-Operator (`=`) ein L-Value zurückgibt

[3] Wurde in ▸ Lektion 02 erklärt

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. und 4. können nicht evaluiert werden, da sie keine validen Expressions sind

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. und 4. können nicht evaluiert werden, da sie keine validen Expressions sind
- 1. evaluiert zu 6, also das Ergebnis der simplen Multiplikation

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. `1*(2*3)`
2. `(a=1)`
3. `(1`
4. `(a*3) = (b*5)`

Lösungen

- 3. und 4. können nicht evaluiert werden, da sie keine validen Expressions sind
- 1. evaluiert zu 6, also das Ergebnis der simplen Multiplikation
- 2. evaluiert zu 1, da `a` zurückgegeben wird, dem gerade davor der Wert 1 zugewiesen wurde

Fragen/Unklarheiten?

6. Tipps zu [code]expert

Tipps für [code] expert

- Früh genug anfangen
- Zusammen daran arbeiten (aber nicht abschreiben!)

7. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche!