



Rückblick

VL Informatik

D-MATH / D-PHYS

HS 2017

Vorlesung 2: Zimtsterne

Ganze Zahlen, Arithmetische Ausdrücke



Wie viele Zimtsterne wollen Sie backen?

11

Zutaten: 2 Eiweiss, 0.55 EL Zimt.

Beispiel zeigt auch

- Konversion `bool` → `int`
- Fliesskommazahlen

```
// Zimtsterne
#include<iostream>

// Zutaten fuer 10 Zimtsterne (Betty Bossi):
// 1 Eiweiss    (unsigned int)
// 0.5 EL Zimt  (float)
// ...

int main()
{
    std::cout << "Wie viele Zimtsterne wollen Sie backen?\n";
    unsigned int zimtsterne;
    std::cin >> zimtsterne;

    std::cout << "Zutaten: ";
    std::cout << zimtsterne / 10 + (zimtsterne % 10 != 0) << " Eiweiss, ";
    std::cout << zimtsterne * 0.05 << " EL Zimt.\n";

    return 0;
}
```

Wie viele Zimtsterne wollen Sie backen?
11
Zutaten: 2 Eiweiss, 0.55 EL Zimt.

Vorlesung 3: Samichlaus-Checker

Typ **bool**, Logische Operatoren, **if-else**



War Ihr Kind brav?
0
Wie alt ist Ihr Kind?
5
Strafe vom Schmutzli!

```
// Samichlaus-Checker
#include<iostream>

int main()
{
    std::cout << "War Ihr Kind brav?\n";
    bool brav;
    std::cin >> brav;

    std::cout << "Wie alt ist Ihr Kind?\n";
    unsigned int alter;
    std::cin >> alter;

    if (brav || alter < 3)
        std::cout << "Geschenk vom Samichlaus!\n";
    else
        std::cout << "Strafe vom Schmutzli!\n";

    return 0;
}
```

```
War Ihr Kind brav?
0
Wie alt ist Ihr Kind?
5
Strafe vom Schmutzli!
```

Vorlesung 3: Ho-Ho-Ho-Automat **for-, while-**-Anweisung



Beispiel zeigt auch
• Postdekrement

Ho Ho Ho
Ho Ho Ho

```
// Ho-Ho-Ho-Automat
#include<iostream>

int main()
{
    for (int i=0; i<3; ++i)
        std::cout << "Ho ";
    std::cout << '\n';

    // Nochmal, weil's so schön war...
    int wie_oft_noch = 3;
    while (wie_oft_noch-- > 0)
        std::cout << "Ho ";
    std::cout << '\n';

    return 0;
}
```

Ho Ho Ho
Ho Ho Ho

Vorlesung 4: Adventskalender do-Anweisung



Welches Türchen?

1

Schööön!

Welches Türchen?

2

Schööön!

Welches Türchen?

4

Falsches Türchen!

Welches Türchen?

```
// Adventskalender
#include<iostream>
#include<vector>

int main()
{
    std::vector<bool> offen (25, false);
    offen[0] = true; // sentinel, Wächter

    unsigned int tag;
    do {
        std::cout << "Welches Türchen?\n";
        std::cin >> tag;
        if (tag < 25 && !offen[tag] && offen[tag-1]) {
            // tag == 0 oder >= 25: Kurzschlussauswertung wichtig!
            offen[tag] = true;
            std::cout << " Schööön!\n";
        } else
            std::cout << " Falsches Türchen!\n";
    } while (!offen[24]);
    return 0;
}
```

Welches Türchen?
1
Schööön!
Welches Türchen?
2
Schööön!
Welches Türchen?
4
Falsches Türchen!
Welches Türchen?

Vorlesung 4: Die Weihnachtsformel

Typen **float** und **double**



$$\text{Number of ornaments} = \frac{\sqrt{17}}{20} \times (\text{Tree height in cms})$$

$$\text{Length of tinsel (cms)} = \frac{13 \times \pi}{8} \times (\text{Tree height in cms})$$

$$\text{Length of lights (cms)} = \pi \times (\text{Tree height in cms})$$

$$\text{Height of the star/angel (cms)} = \frac{(\text{Tree height in cms})}{10}$$

```
// Die Weihnachtsformel
#include<iostream>

int main()
{
    const double nikolaus_tag      = 6.0;
    const double heilige_koenige = 3.0;
    const double adventssonntage = 4.0;

    double x = nikolaus_tag;
    double w = 0.0;
    while (x > 1e-5) {
        std::cout << w << '\n';
        w += x;
        x *= heilige_koenige / adventssonntage;
    }

    return 0;
}
```

```
// Die Weihnachtsformel
#include<iostream>

int main()
{
    const double nikolaus_tag      = 6.0;
    const double heilige_koenige = 3.0;
    const double adventssonntage = 4.0;

    double x = nikolaus_tag;
    double w = 0.0;
    while (x > 1e-5) {
        std::cout << w << '\n';
        w += x;
        x *= heilige_koenige / adventssonntage;
    }

    return 0;
}
```

Was ist die letzte Ausgabe ?

- (a) 7
- (b) 12
- (c) 24
- (d) 42
- (e) 2017

```

// Die Weihnachtsformel
#include<iostream>

int main()
{
    const double nikolaus_tag      = 6.0;
    const double heilige_koenige = 3.0;
    const double adventssonntage = 4.0;

    double x = nikolaus_tag;
    double w = 0.0;
    while (x > 1e-5) {
        std::cout << w << '\n';
        w += x;
        x *= heilige_koenige / adventssonntage;
    }

    return 0;
}

```

Was ist die letzte Ausgabe ?

- (a) 7
- (b) 12
- (c) 24
- (d) 42
- (e) 2017

Lösung: $6 \cdot \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i = 24$

Vorlesung 5: Schnee

Fliesskommazahlen, Funktionen



```
// Schnee
#include<iostream>
#include<cassert>

// PRE: 0 <= d < 1
// POST: gibt die binären Nachkommstellen von d aus,
//        wobei {0,1} durch {' ', '*' } dargestellt werden
void schnee (double d)
{
    while (d != 0) {
        assert (0.0 <= d && d < 1.0);
        d *= 2.0;
        if (d >= 1.0) {
            std::cout << '*';
            d -= 1.0;
        } else
            std::cout << ' ';
    }
    std::cout << '\n';
}

int main()
{
    const double frau_holle = 0.01812;
    for (double d = 0.1; d < 1.0; d += frau_holle)
        schnee (d);
    return 0;
}
```

Vorlesung 6: Weihnachtsbaum Funktionen, Stepwise Refinement



```

// Weihnachtsbaum
#include<iostream>
#include<cassert>
#include<string>

// PRE: total - baum ist gerade
// POST: zeichnet einen Weihnachtsbaumschnitt mit
//        Breite total und Baubreite baum (in der
//        Mitte des Schnitts)
void schnitt (unsigned int total, unsigned int baum)
{
    assert ((total - baum) % 2 == 0);
    std::cout << std::string ((total-baum)/2, ' ');
    std::cout << std::string (baum, '*');
    std::cout << std::string ((total-baum)/2, ' ');
    std::cout << "\n";
}

// PRE: hoehe ist gerade
// POST: zeichnet einen Weihnachtsbaum mit
//        hoehe vielen Schnitten; jeder
//        zweite hat Baubreite 1
void weihnachtsbaum (unsigned int hoehe)
{
    assert (hoehe % 2 == 0);
    for (int i=1; i<hoehe; i+=2) {
        schnitt (hoehe+1, i);
        schnitt (hoehe+1, 1);
    }
}

```

```
int main()
{
    weihnachtsbaum(20);
    return 0;
}

*****
```

Vorlesung 7: Weihnachtsmann

Referenztypen



Ich glaube an Symbolfigur weihnachtlichen Schenkens, Symbolfigur weihnachtlichen Schenkens, Symbolfigur weihnachtlichen Schenkens.

```
// Weihnachtsmann
#include<iostream>
#include<string>

int main()
{
    std::string weihnachtsmann;
    std::string& santa_claus = weihnachtsmann;
    std::string& vaeterchen_frost = weihnachtsmann;
    std::string& usw = weihnachtsmann;

    usw = "Symbolfigur weihnachtlichen Schenkens";

    std::cout << "Ich glaube an "
        << weihnachtsmann << ", "
        << santa_claus << ", "
        << vaeterchen_frost << ".\n";

    return 0;
}
```

Ich glaube an Symbolfigur weihnachtlichen Schenkens, Symbolfigur weihnachtlichen Schenkens, Symbolfigur weihnachtlichen Schenkens.

Vorlesung 7: Einmal werden wir noch wach

Call-by-Reference

Morgen, Kinder, wird's was geben

Text: Martin Friedrich Philipp Bartsch (1795)

Melodie: Carl Gottlieb Hering (1809)

A musical score in G major, common time. The melody starts on G and includes notes for C, G, C, D, G, G, Em, C, G, D. The lyrics for this line are: "1. Mor - gen, Kin - der, wird's was ge - ben, mor - gen wer - den wir uns freun; welch ein Ju - bel, welch ein Le - ben wird in un - serm Hau - se sein!"

A continuation of the musical score in G major, common time. The melody includes notes for Am, C, D, Bm, Em, F, D/F#, G. The lyrics for this line are: "Ein - mal wer - den wir noch wach, hei - ßa, dann ist Weih - nachts - tag!"

2. Wie wird dann die Stube glänzen von der großen Lichterzahl,
schöner als bei frohen Tänzen
ein geputzter Kronensaal!
Wisst ihr noch vom vorgen Jahr,
wie's am Weihnachtsabend war?
3. Wisst ihr noch mein Reiterpferdchen,
Malchens nette Schäferin?
Jettchens Küche mit dem Herdchen
und dem blank geputzten Zinn?
Heinrichs bunten Harlekin
mit der gelben Violin?
4. Wisst ihr noch den großen Wagen und die schöne Jagd von Blei?
Unsre Kleiderchen zum Tragen
und die viele Näscherlei?
Meinen fleißigen Sägemann
mit der Kugel unten dran?
5. Welch ein schöner Tag ist morgen,
viele Freuden hoffen wir!
Unsre lieben Eltern sorgen
lange, lange schon dafür.
O gewiss, wer sie nicht ehrt,
ist der ganzen Lust nicht wert!

Beispiel zeigt auch
• Felder



```
// Einmal werden wir noch wach
#include<iostream>
#include<string>

void swap (std::string& a, std::string& b)
{
    std::string h = a;
    a = b;
    b = h;
}

int main()
{
    std::string lied[] = {"Einmal", "werden", "wach", "noch", "wir"};
    swap (lied[2], lied[4]);
    for (int i=0; i<5; ++i)
        std::cout << lied[i] << ' ';
    std::cout << '\n';
}
```

Vorlesung 8: Weihnachtskugel

Zweidimensionale Felder



```

// Weihnachtskugel
#include<iostream>
#include<cassert>
const unsigned int d = 31; // Durchmesser+1

// PRE: d ist ungerade
void mache_weihnachtskugel (char (&k)[d][d])
{
    assert (d % 2 != 0);
    unsigned int r = d/2; // Radius
    for (int x=0; x<d; ++x)
        for (int y=0; y<d; ++y)
            if ((x-r)*(x-r) + (y-r)*(y-r) < r*r)
                k[x][y] = y % 2 ? '*' : '-';
            else
                k[x][y] = ' ';
}

void drucke_weihnachtskugel (const char (&k)[d][d])
{
    // von oben nach unten
    for (int y=d-1; y>=0; --y) {
        for (int x=0; x<d; ++x)
            // Ostereivermeidung: jedes Zeichen zweimal!
            std::cout << k[x][y] << k[x][y];
        std::cout << '\n';
    }
}

```

Vorlesung 9: Geschenkeliste

Zeiger / Iteratoren, Bereiche



Was wünscht du dir?
iPhone
Nein, das kannst du vergessen!

```
#include<iostream>
#include<string>

// PRE: [begin, end) ist ein gültiger Bereich
// POST: ein Zeiger auf das erste Vorkommen von g in [begin end) wird zurückgegeben;
//        falls g nicht vorkommt, wird end zurückgegeben
const std::string* finde_geschenk (std::string g, const std::string* begin, const std::string* end)
{
    for (const std::string* p = begin; p != end; ++p)
        if (*p == g) return p;
    return end;
}

int main ()
{
    std::string liste[] = {"Barbie", "Holzeisenbahn", "Socken"};

    std::cout << "Was wünscht du dir?\n";
    std::string wunsch;
    std::cin >> wunsch;

    const std::string* p = finde_geschenk (wunsch, liste, liste+3);
    if (p != liste+3)
        std::cout << "Ja, das gibt's vielleicht!\n";
    else
        std::cout << "Nein, das kannst du vergessen!\n";

    return 0;
}
```

Was wünscht du dir?
iPhone
Nein, das kannst du vergessen!

Vorlesung 9: Nüsse verteilen

Zeiger / Iteratoren, Bereiche, Rekursion



Möglichkeiten für 3 Kinder und 6 Nüsse:

0	0	6
0	1	5
0	2	4
0	3	3
0	4	2
0	5	1
0	6	0
1	0	5
1	1	4
1	2	3
1	3	2
1	4	1
1	5	0
2	0	4
2	1	3
2	2	2
2	3	1
2	4	0
3	0	3
3	1	2
3	2	1
3	3	0
4	0	2
4	1	1
4	2	0
5	0	1
5	1	0
6	0	0

```

// Nüsse verteilen          0 0 6
#include<iostream>          0 1 5
                                0 2 4
                                0 3 3
                                0 4 2
                                0 5 1
                                0 6 0
                                1 0 5
                                1 1 4
                                1 2 3
                                1 3 2
                                1 4 1
                                1 5 0
                                2 0 4
                                2 1 3
                                2 2 2
                                2 3 1
                                2 4 0
                                3 0 3
                                3 1 2
                                3 2 1
                                3 3 0
                                4 0 2
                                4 1 1
                                4 2 0
                                5 0 1
                                5 1 0
                                6 0 0

// PRE: [begin, end) ist ein gültiger Bereich, begin <= bedient_end <= end
// POST: alle Möglichkeiten, noch n Nüsse unter den Kindern [bedient_end, end)
//       zu verteilen, werden ausgegeben
void verteile_nuesse (unsigned int n, unsigned int* begin,
                      unsigned int* bedient_end, unsigned int* end)
{
    if (bedient_end == end) { // alle Kinder bedient?
        if (n == 0) { // alle Nüsse verteilt?
            for (unsigned int* p = begin; p != end; ++p)
                std::cout << *p << ' '; // Verteilung ausgeben
            std::cout << '\n';
        }
    } else { // erstes unbedientes Kind bekommt k Nüsse
        for (unsigned int k=0; k<=n; ++k) {
            *bedient_end = k;
            ???
        }
    }
}

```

a) verteile_nuesse (n-k, begin+1, bedient_end, end) 4 1 1

b) verteile_nuesse (n-k, begin, bedient_end-1, end) 4 2 0

c) verteile_nuesse (n-k, begin, bedient_end+1, end) 5 0 1

d) verteile_nuesse (n-k, begin, bedient_end, end-1) 5 1 0

```

// Nüsse verteilen          0 0 6
#include<iostream>          0 1 5
                                0 2 4
                                0 3 3
                                0 4 2
                                0 5 1
                                0 6 0
                                1 0 5
                                1 1 4
                                1 2 3
                                1 3 2
                                1 4 1
                                1 5 0
                                2 0 4
                                2 1 3
                                2 2 2
                                2 3 1
                                2 4 0
                                3 0 3
                                3 1 2
                                3 2 1
                                3 3 0
                                4 0 2
                                4 1 1
                                4 2 0
                                5 0 1
                                5 1 0
                                6 0 0

// PRE: [begin, end) ist ein gültiger Bereich, begin <= bedient_end <= end
// POST: alle Möglichkeiten, noch n Nüsse unter den Kindern [bedient_end, end)
//        zu verteilen, werden ausgegeben
void verteile_nuesse (unsigned int n, unsigned int* begin,
                      unsigned int* bedient_end, unsigned int* end)
{
    if (bedient_end == end) { // alle Kinder bedient?
        if (n == 0) { // alle Nüsse verteilt?
            for (unsigned int* p = begin; p != end; ++p)
                std::cout << *p << ' '; // Verteilung ausgeben
            std::cout << '\n';
        }
    } else { // erstes unbedientes Kind bekommt k Nüsse
        for (unsigned int k=0; k<=n; ++k) {
            *bedient_end = k;
            verteile_nuesse (n-k, begin, bedient_end+1, end);
        }
    }
}

```

```
// PRE: [begin, end) ist ein gültiger Bereich          0 0 6
// POST: alle Möglichkeiten, n Nüsse unter den Kindern [begin, end)   0 1 5
//        zu verteilen, werden ausgegeben           0 2 4
void verteile_nuesse (unsigned int n, unsigned int* begin, unsigned int* end) 0 3 3
{
    verteile_nuesse (n, begin, begin, end);          0 4 2
}
                                                0 5 1
                                                0 6 0
                                                1 0 5
                                                1 1 4
                                                1 2 3
int main()
{
    unsigned int kinder[3];
    verteile_nuesse (6, kinder, kinder+3);          1 3 2
    return 0;                                         1 4 1
}
                                                1 5 0
                                                2 0 4
                                                2 1 3
                                                2 2 2
                                                2 3 1
                                                2 4 0
                                                3 0 3
                                                3 1 2
                                                3 2 1
[3 3 0
4 0 2
4 1 1
4 2 0
5 0 1
5 1 0
6 0 0
```

Vorlesung 10: Kutsche

EBNF / Parsen



Gib eine Kutsche ein!
YYYYYY_
Rentierstärken = 6

```
// Kutsche
#include<iostream>
#include<istream>
#include<string>
#include<sstream>
#include<cassert>

// EBNF für Weihnachtsmann-Kutschen
// -----
// Kutsche = Rentiere "_"
// Rentiere = "Y" {"Y"}

// declarations

// PRE: is = Kutsche...
// POST: Kutsche wird aus is extrahiert und die Anzahl
//       der Rentiere zurückgegeben
unsigned int Kutsche (std::istream& is);

// PRE: is = Kutsche...
// POST: Rentiere wird aus is extrahiert und die Anzahl
//       der Rentiere zurückgegeben
unsigned int Rentiere (std::istream& is);
```

```
// definitions

// POST: leading whitespace characters are extracted
//        from is, and the first non-whitespace character
//        is returned (0 if there is no such character)
char lookahead (std::istream& is)
{
    is >> std::ws;           // skip whitespaces
    if (is.eof())
        return 0;             // end of stream
    else
        return is.peek();     // next character in is
}

// POST: if next chararcer in is is ch, consume c and return
//        true, otherwise return false
bool consume (std::istream& is, char c)
{
    if (lookahead (is) == c) {
        is >> c;
        return true;
    } else
        return false;
}
```

```
unsigned int Kutsche (std::istream& is)
{
    unsigned int r = Rentiere (is);
    if (!consume (is, '_'))
        assert (false); // kein Schlitten
    return r;
}
```

```
unsigned int Rentiere (std::istream& is)
{
    unsigned int r = 0;
    while (consume (is, 'Y'))
        ++r;
    assert (r > 0); // mindestens ein Rentier
    return r;
}
```

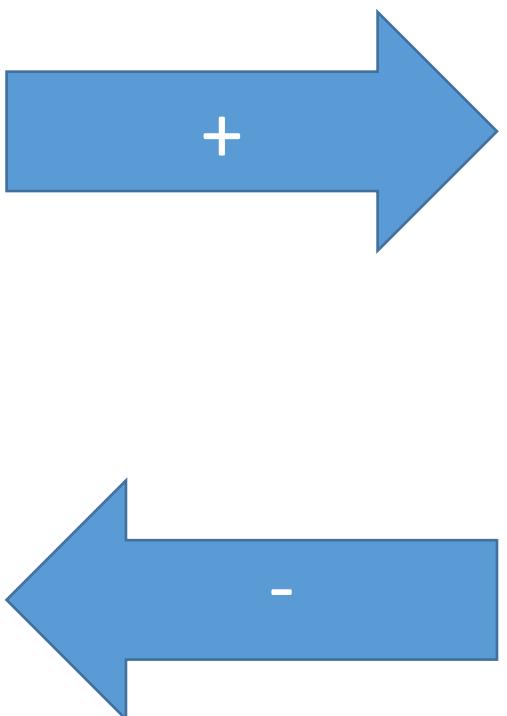
```
int main()
{
    std::cout << "Gib eine Kutsche ein!\n";
    std::string k;
    std::cin >> k;
    std::stringstream santamobil (k);
    std::cout << "Rentierstärken = " << Kutsche (santamobil) << '\n';

    return 0;
}
```

Gib eine Kutsche ein!
YYYYYY_
Rentierstärken = 6

Vorlesung 11: Zaubersack

Structs, Operatoren



```

// Zaubersack
#include<iostream>

struct sack {
    unsigned int geschenke; // Kapazität
};

// POST: vergrößert s um den Faktor 2
sack& operator+ (sack& s)
{
    s.geschenke *= 8;
    return s;
}

// POST: verkleinert s um den Faktor 2
sack& operator- (sack& s)
{
    s.geschenke /= 8;
    return s;
}

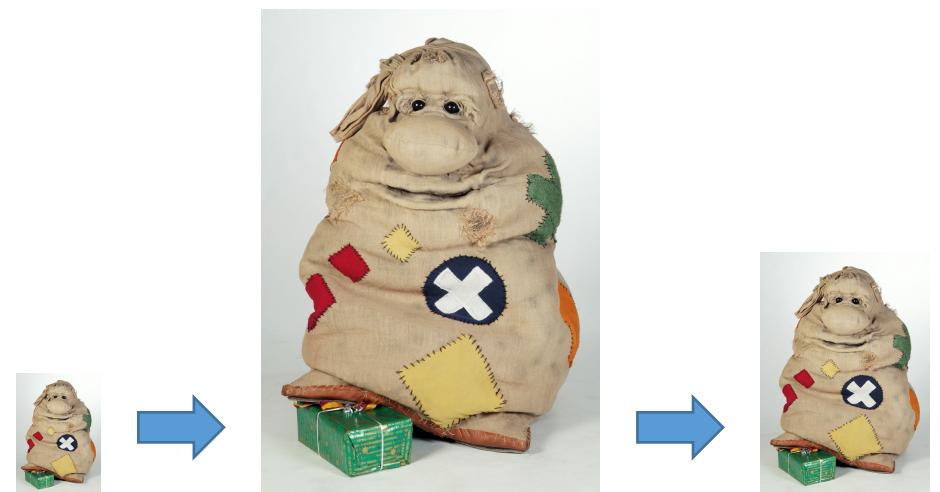
```

```

int main()
{
    sack s = {3};
    +(s);
    std::cout << s.geschenke
                << '\n'; // 196
    -s;
    std::cout << s.geschenke
                << '\n'; // 24

    return 0;
}

```



Vorlesung 12: Rudolph Klassen, Members, Konstruktoren



```
// Rudolph
#include<iostream>
#include<string>

class Rentier
{
public:
    // Default-Konstruktor
    Rentier()
        : nasenfarbe ("braun")
    {}

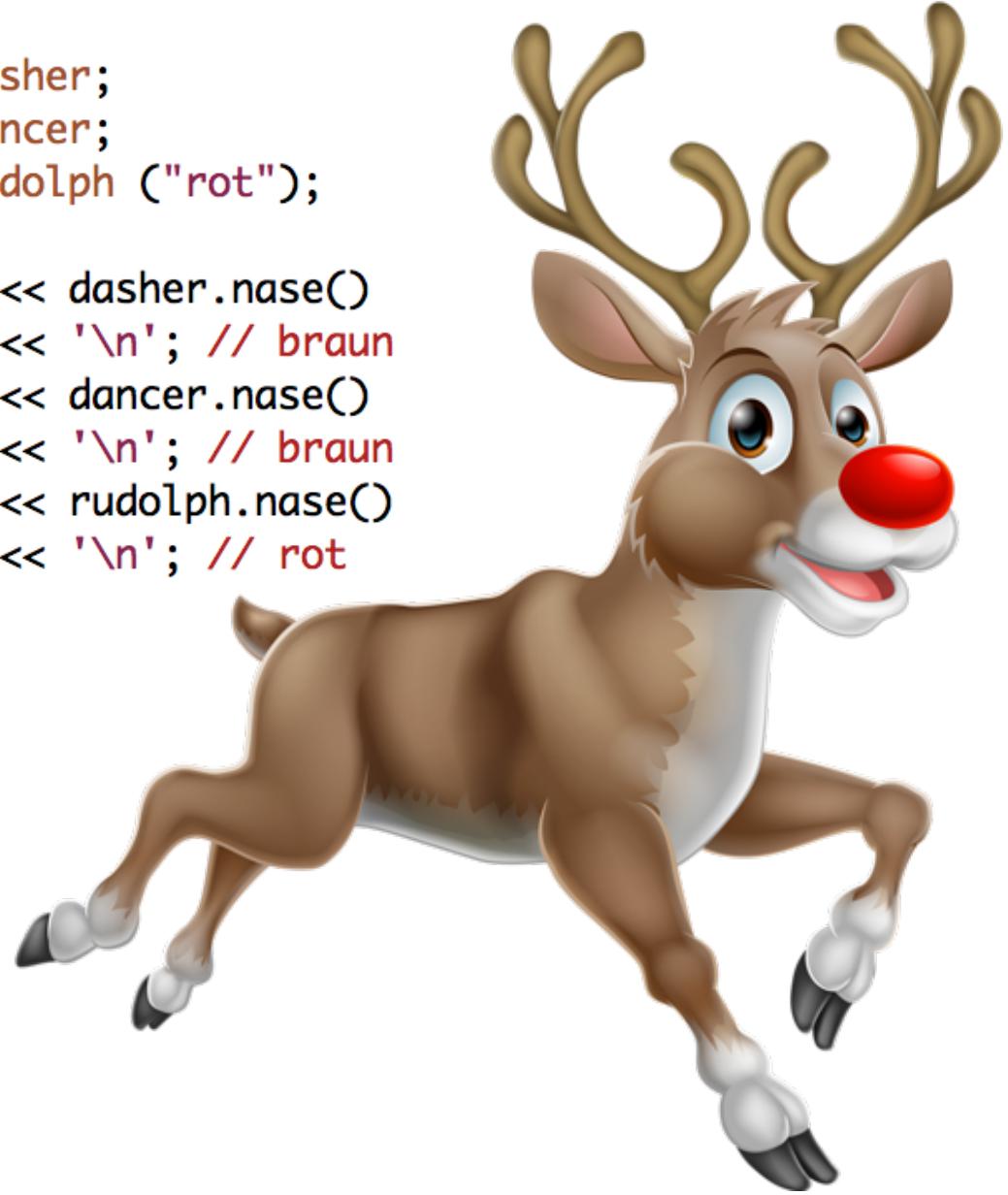
    // Konstruktor mit Farbe
    Rentier (std::string f)
        : nasenfarbe (f)
    {}

    // Nasenfarbe
    std::string nase () const
    {
        return nasenfarbe;
    }
private:
    std::string nasenfarbe;
};
```

```
int main ()
{
    Rentier dasher;
    Rentier dancer;
    Rentier rudolph ("rot");

    std::cout << dasher.nase()
                  << '\n'; // braun
    std::cout << dancer.nase()
                  << '\n'; // braun
    std::cout << rudolph.nase()
                  << '\n'; // rot

    return 0;
}
```



Vorlesung 12: Bescherung

Dynamische Datentypen



```
// Bescherung
#include<iostream>
#include<string>

class Geschenk {
private:
    std::string* was_ist_es;
public:
    // Inhalt
    std::string inhalt () const
{
    if (was_ist_es != 0)
        return *was_ist_es;
    else
        return "nichts";
}
```



```
// Default-Konstruktor
Geschenk ()
    : was_ist_es (0)
{
    std::cout << "Da ist ja gar nichts drin!\n";
}

// Konstruktor mit String
Geschenk (const std::string& i)
    : was_ist_es (new std::string (i))
{
    std::cout << "Wow, " << *was_ist_es << ", danke!\n";
}
```

```
// Copy-Konstruktor
Geschenk (const Geschenk& g)
    : was_ist_es (0)
{
    if (g.was_ist_es != 0) {
        was_ist_es = new std::string (*g.was_ist_es);
    }
    std::cout << "Ok, nochmal " << inhalt() << "...\\n";
}

// Destruktor
~Geschenk ()
{
    std::cout << inhalt() << "... das war ein tolles Geschenk!\\n";
    delete was_ist_es;
}
```

```
// Zuweisungs-Operator
Geschenk& operator= (const Geschenk& g)
{
    if (was_ist_es != g.was_ist_es) { // vermeide Selbstzuweisung
        std::cout << "Du tauschst mir " << inhalt() << " gegen ";
        if (was_ist_es != 0) {
            delete was_ist_es;
            was_ist_es = 0;
        }
        if (g.was_ist_es != 0)
            was_ist_es = new std::string (*g.was_ist_es));
        std::cout << inhalt() << " ??\n";
    }
    return *this;
}
}; // Ende Geschenk
```

Bescherung!

```
int main()
{
    Geschenk g1;
    Geschenk g2 ("Barbie");
    Geschenk g3 ("Holzeisenbahn");
    Geschenk g4 ("Socken");
    Geschenk g5 = g4;
    g5 = Geschenk ("iPhone");
}
```

Da ist ja gar nichts drin!
Wow, Barbie, danke!
Wow, Holzeisenbahn, danke!
Wow, Socken, danke!
Ok, nochmal Socken...
Wow, iPhone, danke!
Du tauschst mir Socken gegen iPhone ??
iPhone... das war ein tolles Geschenk!
iPhone... das war ein tolles Geschenk!
Socken... das war ein tolles Geschenk!
Holzeisenbahn... das war ein tolles Geschenk!
Barbie... das war ein tolles Geschenk!
nichts... das war ein tolles Geschenk!