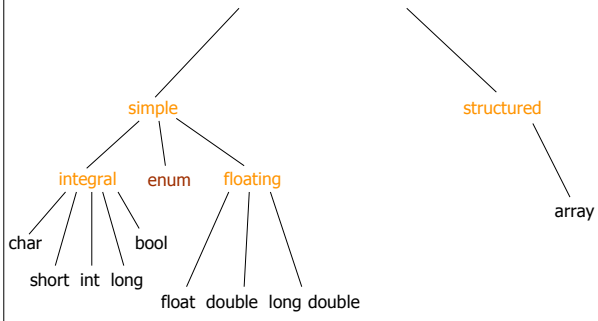
Grundlagen der Programmierung in C++

Zusammengesetzte Typen

Wintersemester 2005/2006
G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de

Das C++ Typsystem



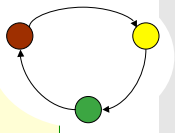
Aufzählungen (enumeration)

- Einfachste Form eines benutzerdefinierten Typs
- Englisch "enumeration" → keyword **enum**
- Bsp.:

```
enum ColorE
{
  RED, GREEN, BLUE
};

ColorE screen_color = BLUE;
ColorE window_color = RED;

int    n = BLUE;        // OK
ColorE c = 1;           // error!
```


Beispiel: Zustandsautomat

- Beispiel Ampel:



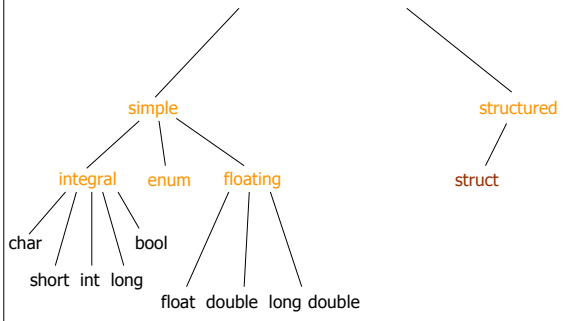
```
enum ColorE { RED, YELLOW, GREEN };

ColorE color; // current state
...
switch ( color )
{
    // switch to next state
    case RED: color = YELLOW;
              break;
    case YELLOW: schalte Rot und Gelb aus;
                 color = GREEN;
                 break;
    case GREEN: schalte Grün aus;
                color = RED;
                break;
    default: fprintf(stderr, "BUG ...");
}
schalte color ein;
...
```

- Was *nicht* geht mit enum's in C++ (ging in C):
  - Rechnen, z.B. `enum++`
- Bestimmung der Anzahl der Enum-Namen:

```
enum ColorE
{
    RED,
    YELLOW,
    GREEN,
    NUM_COLORS
};
```

  - Funktioniert, weil:
    - Enum-Namen werden fortlaufend numeriert
    - Erster Enum-Name = 0

Das C++ Typsystem



Struct

- Elemente verschiedenen Typs zusammenfassen zu einem Neuen
- Beispiel: Name, Alter, Größe, …, einer Person
- Sprach-Feature: "zusammengesetzter Typ"
- Definition:

```
struct NewName
{
    T1 M1;
    T2 M2;
    ...
};
```

wobei `T1, T2`, …, bekannte Typen sind,
und `M1, M2`, …, die sogenannten *Member*.

- Beispiel:

```
struct Person
{
    unsigned int ID;
    unsigned int age;
    float        salary;
    ColorE       color;
};
Person person;
Person p2 = { 007, 99, 1000000.0, black };
```

- Benutzung:

```
p2 = person;                // assignment
person.age = 17;            // member access
printf("%f ", person.color );
```

Member access operator

- Mathematisches Äquivalent: Produkt

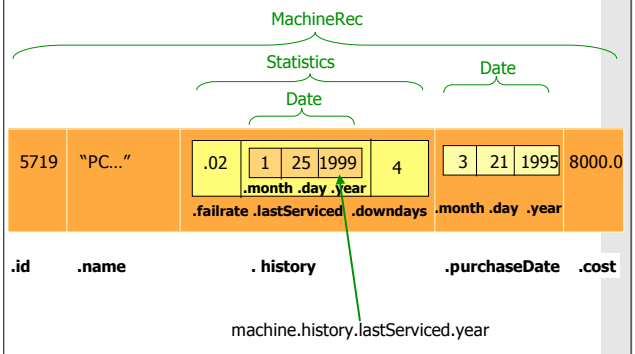
  Sei S Struct mit Members der Typen T_1, ..., T_n;
  dann ist der Wertebereich von S

$$W(S) = \prod_i W(T_i)$$

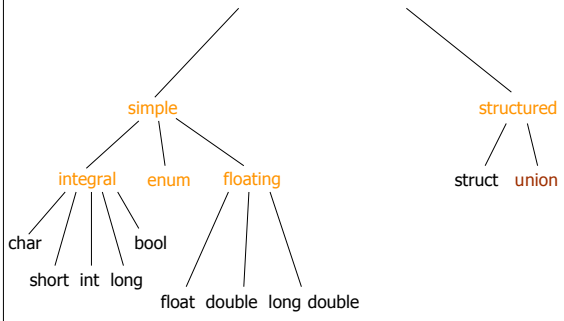
Komposition

- Structs können wieder Structs enthalten
- Vorteil: "Modularisierung" der Datentypen
- Beispiel: Maschinen-Record
  - Datum der Anschaffung
  - Preis
  - Name
  - Statistik der Ausfälle


```
struct  Date
{   int    month;
    int    day;
    int    year;
};

struct  Statistics
{   float    failRate;
    Date     lastServiced;          //  composition
    int      downDays;
};

struct  MachineRec
{   int            id;
    string         name;
    Statistics     history;        //  composition
    Date           purchaseDate;   //  composition
    float          cost;
};

MachineRec  machine;
machine.history.lastServiced.year = 1999;
```

Das C++ Typsystem



Union

- Enthält *Alternativen*
- Definition:

```
union NewTypeName
{
    T1 M1;
    T2 M2;
    ...
};
```

  wobei `T1`, `T2`, …, bekannte Typen sind,
  und `M1`, `M2`, …, die sogenannten *Member*.
- Hauptsächliche Verwendung: Speicherplatz einsparen

- Beispiel:

```
union IntOrFloat
{
    int   i;
    float f;
};
IntOrFloat iof;
iof.i = 10;
printf("%f", iof.f );      // ergibt Bogus
```

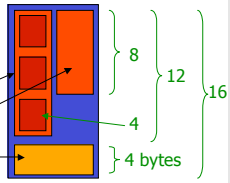
- Achtung: Union "weiß" nicht, welcher Member gültig ist!
- Lösung: Tag
  - Kombiniere Union mit Enum in Struct
  - Enum zeigt an, welcher Member des Union gültig

Beispiel für Tag und anonymen Union

```
struct DayMonthYear
{   unsigned int    day;
    unsigned int    month;
    unsigned int    year;
};

enum DateFormatE { DAY_MONTH_YEAR, SECONDS };

struct Date
{
    union                                   <-- Anonymer Union
    {
        DayMonthYear  date;
        long long int seconds; // since 1.1.1970
    };
    DateFormatE format;                     <-- Tag (discriminator)
};
Date date;
date.format = DAY_MONTH_YEAR;
Date.date.year = 1999;
```

- Speicherlayout dieser Datenstruktur:
  - Annahme: enum = 4 bytes, int = 4 bytes, long long = 8 bytes

```
struct DayMonthYear
{   unsigned int    day;
    unsigned int    month;
    unsigned int    year;
};

struct Date
{
    union
    {
        DayMonthYear  date;
        long long int seconds;
    };
    DateFormatE format;
};
```



- Mathematisches Äquivalent: Fallunterscheidung
  Sei U Union mit Members der Typen T_1, …, T_n;
  dann ist der Wertebereich von U

$$W(U) = \begin{cases} W(T_1) & \text{falls Tag} = t_1 \\ \vdots & \\ W(T_n) & \text{falls Tag} = t_n \end{cases}$$

Beispiel für *Weak Typing* in C++

- Weak Typing = Sprache, die Uminterpretieren von Speicherinhalten erlaubt (is a Bad Thing)

```
union
{
    int i;
    float f;
} int_and_float;

int_and_float x;
x.f = 10.0;

if ( x.i & 0x80000000 )
    // x.f ist negativ
```