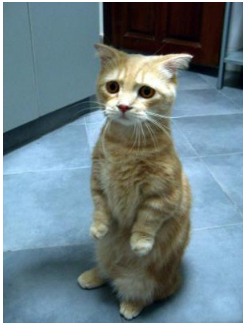
Learn You a Haskell
for Great Good!
Learn u a Haskell for great Good!

- **Nicht:** Anweisungen, die nacheinander ausgeführt werden.

- Nicht: Anweisungen, die nacheinander ausgeführt werden.
- „Besteht ein Programm die Typprüfung, so ist es korrekt.“

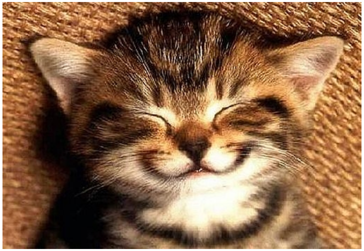
# Quicksort in C

```c
// von rosettacode.org
void quick_sort (int *a, int n) {
    int i, j, p, t;
    if (n < 2)
        return;
    p = a[n / 2];
    for (i = 0, j = n - 1;; i++, j--) {
        while (a[i] < p)
            i++;
        while (p < a[j])
            j--;
        if (i >= j)
            break;
        t = a[i];
        a[i] = a[j];
        a[j] = t;
    }
    quick_sort(a, i);
    quick_sort(a + i, n - i);
}
```



# Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```



# Haskell, eine rein funktionale Sprache

```haskell
qsort []       = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

# Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rl} \texttt{fibs} = & \boxed{1} : \boxed{1} : ?? \\ \texttt{tail fibs} = & \boxed{1} : ?? \\ \hline + & \boxed{2} : ?? \end{array}$$

## Haskell, eine rein funktionale Sprache

```haskell
qsort []      = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rl} \texttt{fibs} = & \boxed{1} : \boxed{1} : \boxed{2} : ?? \\ \texttt{tail fibs} = & \boxed{1} : ?? \\ \hline + & \boxed{2} : ?? \end{array}$$

# Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rl}
\texttt{fibs} = & \boxed{1} : \boxed{1} : \boxed{2} : ?? \\
\texttt{tail fibs} = & \boxed{1} : \boxed{2} : ?? \\
\hline
+ & \boxed{2} : ??
\end{array}$$

## Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rl}
\texttt{fibs} = & \boxed{1} : \boxed{1} : \boxed{2} : ?? \\
\texttt{tail fibs} = & \boxed{1} : \boxed{2} : ?? \\
\hline
+ & \boxed{2} : \boxed{3} : ??
\end{array}$$

# Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rl} \texttt{fibs} = & \boxed{1} : \boxed{1} : \boxed{2} : \boxed{3} : ?? \\ \texttt{tail fibs} = & \boxed{1} : \boxed{2} : ?? \\ \hline + & \boxed{2} : \boxed{3} : ?? \end{array}$$

# Haskell, eine rein funktionale Sprache

```haskell
qsort []      = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rl} \texttt{fibs} = & \boxed{1} : \boxed{1} : \boxed{2} : \boxed{3} : ?? \\ \texttt{tail fibs} = & \boxed{1} : \boxed{2} : \boxed{3} : ?? \\ \hline + & \boxed{2} : \boxed{3} : ?? \end{array}$$

# Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rl} \texttt{fibs} = & \boxed{1} : \boxed{1} : \boxed{2} : \boxed{3} : ?? \\ \texttt{tail fibs} = & \boxed{1} : \boxed{2} : \boxed{3} : ?? \\ \hline + & \boxed{2} : \boxed{3} : \boxed{5} : ?? \end{array}$$

# Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

$$\begin{array}{rcccccccccccccc}
\texttt{fibs} = & \boxed{1} & : & \boxed{1} & : & \boxed{2} & : & \boxed{3} & : & \boxed{5} & : & \boxed{8} & : & \boxed{13} & : \ldots \\
\texttt{tail fibs} = & \boxed{1} & : & \boxed{2} & : & \boxed{3} & : & \boxed{5} & : & \boxed{8} & : & \boxed{13} & : & \boxed{21} & : \ldots \\
\hline
+ & \boxed{2} & : & \boxed{3} & : & \boxed{5} & : & \boxed{8} & : & \boxed{13} & : & \boxed{21} & : & \boxed{34} & : \ldots
\end{array}$$

## Haskell, eine rein funktionale Sprache

```haskell
qsort []     = []
qsort (x:xs) =
    qsort kleinere ++ [x] ++ qsort groessere
    where
    kleinere  = [y | y <- xs, y <= x]
    groessere = [y | y <- xs, y > x]
```

Die Fibonaccizahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```haskell
fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
```

**♥ Statisches Typsystem mit Typerschließung ♥**

**rein funktional • nebenläufig • lazy • 7000$^{+}$ Pakete**