Funktionen und Scopes

Codebeispiele

Was macht der folgende Code?

```
int a=4, b=7; // input
int x=-1; // output
if (a >= b) {
  x = a;
} else {
  x = b;
}
```

Codebeispiele

Er berechnet das Maximum

```
int a = 4, b = 7; // input
int x = max(a, b); // output
```

Codebeispiele

Was macht der folgende Code?

```
int a = -3; // or 4
int x = 0; // output
if (a < 0) {
  x = -a;
} else {
  x = a;
}
```

Codebeispiele

Er berechnet den Betrag!

```
int a = -3; // or 4
int x = abs(a);
```

Codebeispiele

Was macht der folgende Code?

```
int a = 4, b = 7; // input and output
int tmp = a;
a = b;
b = tmp;
```

Codebeispiele

Er vertauscht die beiden Werte!

```
int a = 4, b = 7; // input and output
swap(a, b);
```

Funktionen

Definition:

```
int square(int x) {
  return x*x;
}
```

Benutzung:

```
int x = square(3);
int y = square(x);
if (square(x) == square(y)) { ... }
```

Funktionen

```
int square(int x) {...}
 |     |     -----  -----
 |    name     |       |
 |             |    body
return type    |
           parameter
```

Ein Parameter hat einen Typ und einen Namen. Im Rumpf (Body) der Funktion kann sie wie eine Variable benutzt werden.

Der Rückgabewert (return value) der Funktion wird mit einer return-Anweisung angegeben:

```
return x*x;
```

Funktionen

Mehrere Parameter und mehrere Returns sind erlaubt:

```
int max_of_3(int a, int b, int c) {
  if (a > b) {
    if (a > c)
      return a;
    else
      return c;
  } else {
    if (b > c)
      return b;
    else
      return c;
  }
}
```

Funktionen

Andere Funktionen können benutzt werden:

```
int max_of_3(int a, int b, int c) {
  return max(max(a, b), c);
}
```

Funktionen

Manchmal kann nützlich sein, Funktionen ohne Rückgabewert zu definieren (mit Referenzen -- siehe später):

```
void function(...) {
    ...
    return;
}
```

Scopes

Der Scope (Sichtbarkeitsbereich) ist der Bereich zwischen der Definition einer Variable und dem entsprechenden "}".

```
signed main() {
  int a; // scope of a begins
  int b; // scope of b begins
  if (...) {
    int c = a+b; // scope of c begins (and uses a and b)
  } // <-- scope of c ends
  int d = c; // error: c is not in scope
  int e = a; // ok

  a = e+b; // ok
} // <- scopes of a, b and e end
```

Scopes

Es ist nicht erlaubt, zweimal die gleiche Variable zu definieren.

```
signed main() {
  int a = 2;
  int a = 3; // error
}
```

Scopes

Aber in verschachtelten Blöcken ist es ok:

```
signed main() {
  if (...) {
    int a = 3;
  } else {
    int a = 1; // ok
  }
}
```

Scopes

Achtung: Man kann Variablen verdecken!

Das ist häufig ein Fehler.

```
signed main() {
  int a = 1;
  if (...) {
    print(a); // prints 1
    int a = 2;
    print(a); // prints 2
  }
  print(a); // prints 1
}
```

Scopes

Funktionen haben ihren eigenen Scope.

```
int f(int x) {
  return a + x; // error: a doesn't exist
}

signed main() {
  int a;
  print(f(3));
}
```

Scopes

Funktionen sind auch Namen.

```
signed main() {
  int a;
  print(f(3));  // error, f is not declared
}

int f(int x) {
  return 2*x;
}
```

Scopes

For-Schleifen sind ein bisschen speziell:

```
signed main() {
  for (int i = 0; i < 4; ++i) { // def. of i
    // can use i
  }
  // can't use i
}
```

Scopes

Guideline: Halte den Scope einer Variable so klein wie möglich.

Schlecht:

```
int n;
int answer = -1;
n = read_int();
print(answer);
```

Scopes

Guideline: Halte den Scope einer Variable so klein wie möglich.

Gut:

```
int n = read_int();
int answer = n*n;
print(answer);
```

Scopes

Guideline: Halte den Scope einer Variable so klein wie möglich.

Schlecht:

```
int sum = 0;
int x = 0;
for (int i = 0; i < n; ++i) {
  x = read_int();
  sum += x;
}
print(sum);
```

Scopes

Guideline: Halte den Scope einer Variable so klein wie möglich.

Gut:

```
int sum = 0;
for (int i = 0; i < n; ++i) {
  int x = read_int();
  sum += x;
}
print(sum);
```

Nützliche Funktionen

Nutze die von der Standardbibliothek bereitgestellten Funktionen!

```
max(a, b); // maximum of two numbers
min(a, b); // minimum of two numbers
max({a, b, c, d, e}); // max of any amount of numbers
min({a, b, c, d, e}); // min of any amount of numbers
gcd(a, b); // greatest common divisor of a and b
abs(x); // absolute value of x

sort(v.begin(), v.end());

swap(a, b); // swaps values of a and b
            // it can do so using references
            // -- wait for advanced c++ lecture
```