

# Functions and Scopes

---

## Code Examples

What is the following code doing?

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

It's computing the maximum!

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

## Code Examples

What is the following code doing?

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

It's computing the absolute value!

```
int a = -3; // or 4
```

```
int x = abs(a);
```

## Code Examples

What is the following code doing?

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

It's swapping the two values

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

# Functions

Definition:

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

Usage:

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



# Functions

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

A parameter has a "type" and a "name". In the body the parameter can be used like a variable.

A function returns a value of the declared type using a "return" instruction:

```
return x*x;
```

# Functions

You can have multiple parameters and multiple returns:

```
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;  
    }  
}
```

You can use other functions too!

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

# Functions

Sometimes, it can be useful to create a function that never returns a value. They are declared using the void type:

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

## Scopes

A scope is the range between the definition of a variable and the corresponding "}".

```
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
```

You can't define the same variable twice.

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

But you can define them in different blocks.

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

# Scopes

Careful: You can hide variables!

Very often this is a mistake.

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



# Scopes

Functions have their own scope too.

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

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

# Scopes

Functions are names too.

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

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

For loops are a bit weird.

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

Guideline: Keep the scope of a variable as small as possible.

Bad:

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

Guideline: Make the scope of a variable as small as possible.

Good:

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

# Scopes

Guideline: Make the scope of a variable as small as possible.

Bad:

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

Guideline: Make the scope of a variable as small as possible.

Good:

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

## Useful functions

Helper functions provided by the standard library. Use them!

```
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
```

```
swap(a, b); // swaps values of a and b
```

```
    // it can do so using references
```

```
    // -- wait for advanced c++ lecture
```