

Creative Programming

Arrays and Functions

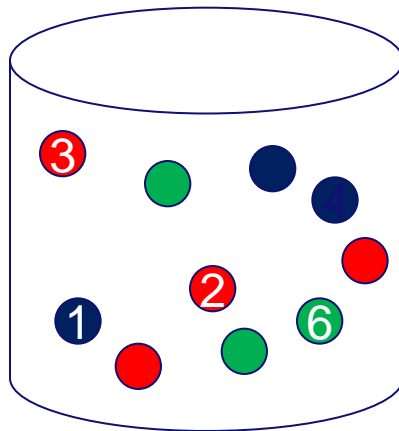
TU / **e**

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Arrays Intro

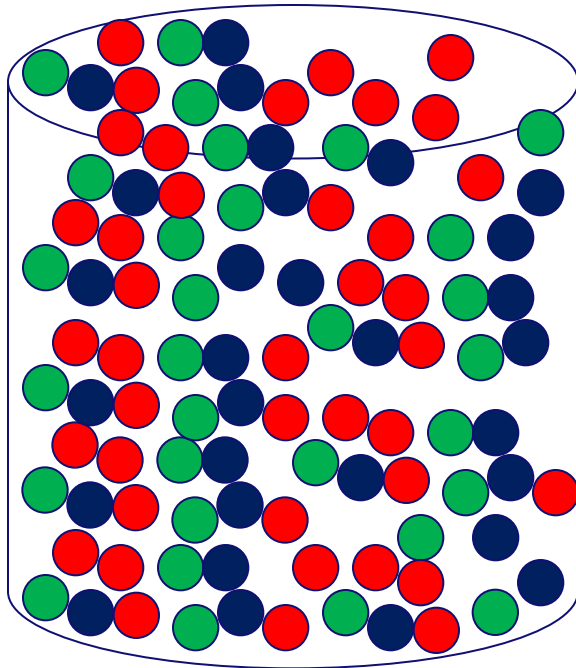
- **Suppose you want to model a bag of 10 balls**
- **Each ball is either red (0) or blue (1) or green (2)**



```
int ball1 = 0;  
int ball2 = 1;  
...  
int ball10 = 2;
```

Arrays Intro

- Now you want to do the same for 100 balls...



```
int ball1 = 0;  
int ball2 = 1;  
...  
int ball100 = 2;
```

100 variables...

- Now you want to do the same for 1000 balls...

Arrays: general idea

- **A variable that is used as a container of variables**
 - Like the bucket of balls
- **Can *hold multiple* values of the *same* type**
- **Access through indexing**
 - `bucket[1] = 0;`
 - `bucket[20] = 2;`
- **Can be of any size***

* But it *is* limited to 2147483647 elements because of indexing with variable of type int and amount of memory

Arrays : types and size

- **Declaration:**

```
int[] bucket;
```

holds ints only!!

reserves space for object

- **Initialization:**

```
bucket = new int[100];
```

holding 100 int's

- **Combining declaration and initialization:**

```
int[] speeds = {56, 34, 93, 120, 5, 54};
```

```
float[] ySpeed = new float[100];
```

```
String[] threeNames = {"Jun", "Loe", "Peter"};
```

Arrays: indexing

- **Index starts counting at “0” !!!!!**
`int firstElement = speeds[0];`
- **length property = actual number of items in the array**
`int len = speeds.length;`
- **Last element is: `speeds[speeds.length-1]`**
- **Runtime error when you go out of range (try it!)**

Looping over an array

```
String[]  
firstNames={"Rene","Loe","Sjriek","Peter"};  
  
for(int i=0; i<=firstNames.length; i++){  
    println(firstNames[i]);  
}
```

Results in?



Examples



Circles



Lots of circles



Sortingcircles

Functions

The background image shows a campus scene with a large yellow sculpture of a hand holding a pen. People are walking and cycling on a path. The image is partially obscured by a semi-transparent orange overlay on the left side.

TU / **e**

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Functions by example

```
int v1, v2, v3, v4, m1, m2, max;
```

```
v1 = 20; v2 = 7; v3 = -1; v4 = 3;
```

```
if (v1 > v2) m1 = v1; else m1 = v2;
```

```
if (v3 > v4) m2 = v3; else m2 = v4;
```

```
if (m1 > m2) max = m1; else max = m2;
```

```
println(max);
```

```
m1 = myMax(v1,v2);
```

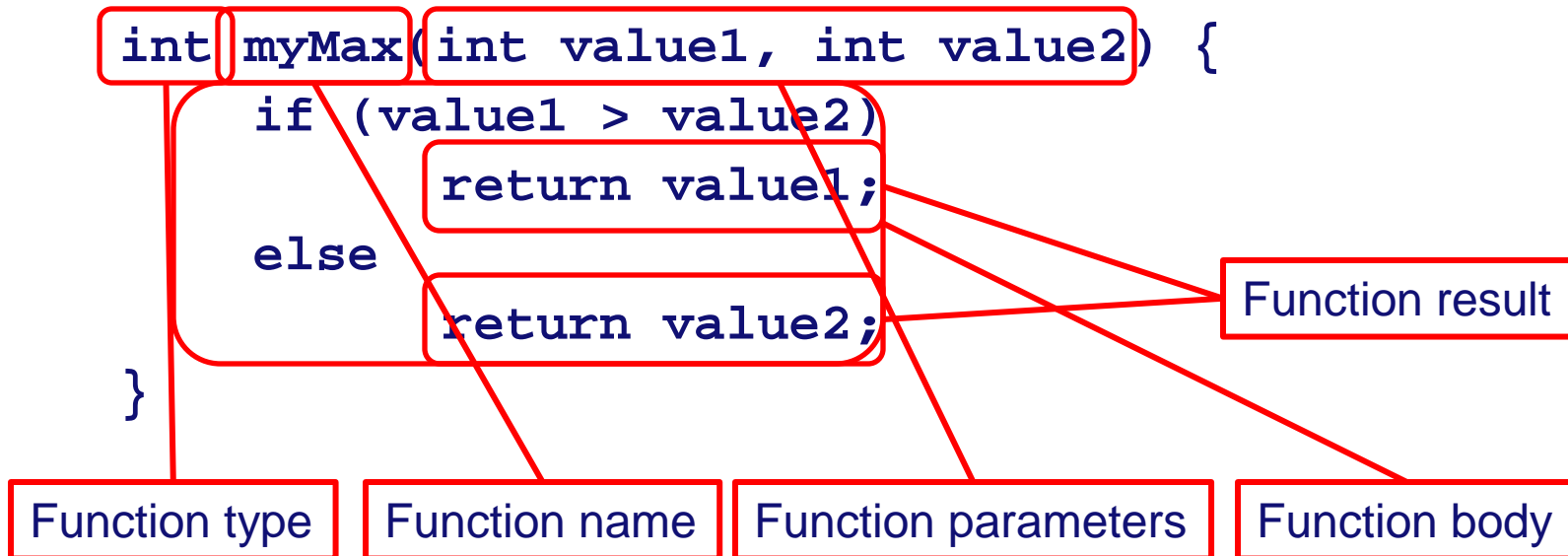
```
m2 = myMax(v3,v4);
```

```
max = myMax(m1,m2);
```



Functions: when, why?

- Functions are reusable blocks of code
- Functions add structure to your program



Functions: how, what?

```
int v1, v2, v3, v4, m1, m2, max;
```

```
v1 = 20; v2 = 7; v3 = -1; v4 = 3;
```

```
m1 = myMax(v1,v2);
```

```
m2 = myMax(v3,v4);
```

```
max = myMax(m1,m2);
```

```
println(max);
```

or

```
println( myMax( myMax(v1,v2), myMax(v3,v4) ) );
```



Functions: Another example

```
void setup(){  
  size(400, 400);  
  background(255);  
  for(int i=0; i<100; i++){  
    rect(  
      int(random(width)), int(random(height)),  
      random(200), random(200)  
    );  
  }  
}
```

Predefined function

- What will happen?



Functions: parameters & arguments

- Parameters passed should match definition of function

```
void myFunction(int x, int y){}
void myFunction(int x, int y, int z){}
void myFunction(float x, float y, float z){}
```

```
myFunction(2, 5);
myFunction(2, 5, 7);
myFunction(2.0, 3.4, 2.33);
```



What does myFunction DO???

Functions: return

- Each function that has a return type other than void *must* return using a return statement

```
int sum;
int computeSum(int x1,int x2,int x3){
    // you could do all sorts of stuff
    // with x1, x2 and x3 before the return
    return x1+x2+x3;
}
sum = computeSum(4,5,6);
```


Summary Arrays and Functions

- **Arrays:**
 - **Collection of similar data objects**
 - **Access via indexing 0..length-1**

- **Functions:**
 - **Grouping of statements that perform a function**
 - **Efficient by avoiding duplicate code**
 - **Frees you from writing linear code**
 - **Enable you to think more abstract**