KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Fun with Functions

Slide	Topic
2-8	Writing Custom Functions

Writing Custom Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Remember: a **function** is like a recipe.

- When you **call** (use) a **function**, the computer (or robot) does all of the actions listed in the "recipe" **in the order they are listed**.
- **Functions** are very helpful if you take some actions multiple times:
  - driving straight forward → **drive_forward();**
  - making a 90° left turn → **turn_left_90();**
  - making a 180° turn → **turn_around();**
  - lifting an arm up → **lift_arm();**
  - closing a claw → **close_claw();**

We made these up… and that's the point!

You can write your own functions to do whatever you want!

- **Functions** often make it easier to **(1)** read the **main** function, and **(2)** change distance, turning, timing, or other values as necessary.

Writing Custom Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

There are **three components** to a function:

1. **Function prototype:** a *promise* to the computer that the function is defined somewhere (like an entry in the table of contents of a recipe book)
2. **Function definition:** the list of actions to be executed (the recipe)
3. **Function call:** using the function (recipe) in your program

Writing Custom Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  #include <kipr/wombat.h>
2
3  void turn_left_90();
4
5  int main()
6  {
7      turn_left_90();
8      return 0;
9  }
10
11 void turn_left_90()
12 {
13     while(gmpc(0) < 1350)
14     {
15         motor(0,100);
16         motor(3,0);
17     }
18     ao();
19 }
```

Function prototypes go above `main`.

Function calls go inside `main` (or inside other functions).

Function definitions go below `main`.

Use `void` in your function prototype if you are ***commanding*** the robot to do something.

Writing Custom Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

The function prototype and the function definition first line *look* the same *except for one thing…*

Source Code

prototype

definition

```
1  #include <kipr/wombat.h>
2
3  void turn_left_90();
4
5  int main()
6  {
7       turn_left_90();
8       return 0;
9  }
10
11 void turn_left_90()
12 {
13      while(gmpc(0) < 1350)
14      {
15           motor(0,100);
16           motor(3,0);
17      }
18      ao();
19 }
```

Notice: no semicolon!
(Why not?)

Writing Custom Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  #include <kipr/wombat.h>
2
3  void turn_left_90();
4
5  int main()
6  {
7       turn_left_90();
8       return 0;
9  }
10
11 void turn_left_90()
12 {
13      while(gmpc(0) < 1350)
14      {
15           motor(0,100);
16           motor(3,0);
17      }
18      ao();
19 }
```

The function prototype is a *promise* to the computer…

… that you will tell the computer *what* to do in the function definition.

Neither the function prototype nor the function definition tell the computer *when* to use the function. That is the job of the function call…

Writing Custom Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  #include <kipr/wombat.h>
2
3  void turn_left_90();
4
5  int main()
6  {
7      turn_left_90();
8      return 0;
9  }
10
11 void turn_left_90()
12 {
13     while(gmpc(0) < 1350)
14     {
15         motor(0,100);
16         motor(3,0);
17     }
18     ao();
19 }
```

The function call makes the computer jump down to the function definition.

The program then executes all of the lines of code in the block of code.

After the computer executes all of the lines of code in the function definition, the program jumps back up to the line of code ***after*** the function call and continues.

Writing Custom Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  #include <kipr/wombat.h>
2  // function prototypes
3  void turn_left();
4  void turn_right();
5
6  int main()
7  {
8        turn_left(); // turn_left function call
9        turn_right(); // turn_right function call
10       return 0;
11 }
12
13 void turn_left() // turn_left function definition
14 {
15       while(gmpc(0) <= 1350)
16       {
17             motor(0,100);
18             motor(3,0);
19       }
20 }
21 void turn_right() // turn_right function definition
22 {
23       while(gmpc(3) <= 1350)
24       {
25             motor(3,100);
26             motor(0,0);
27       }
28       ao();
29 }
```