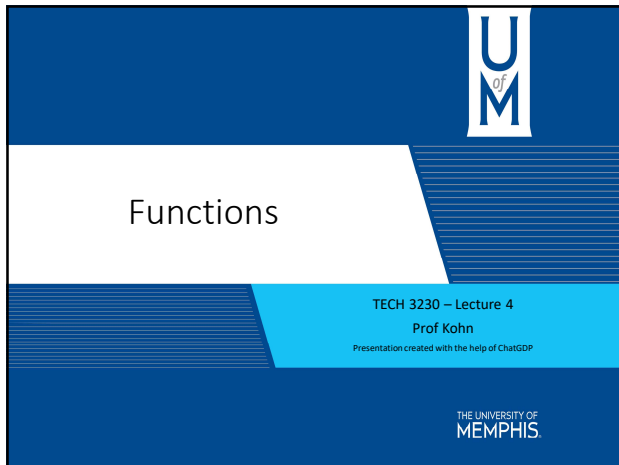
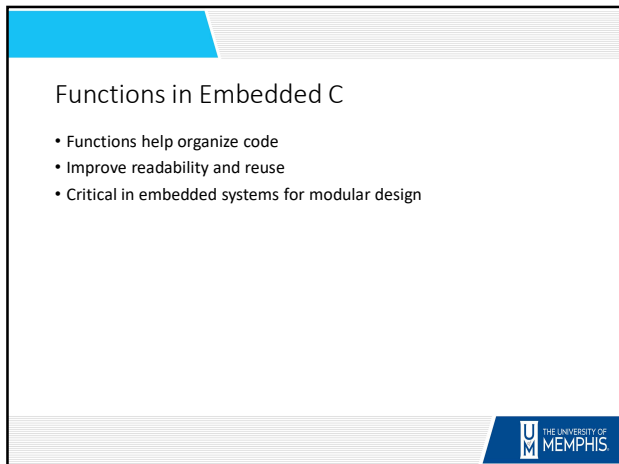
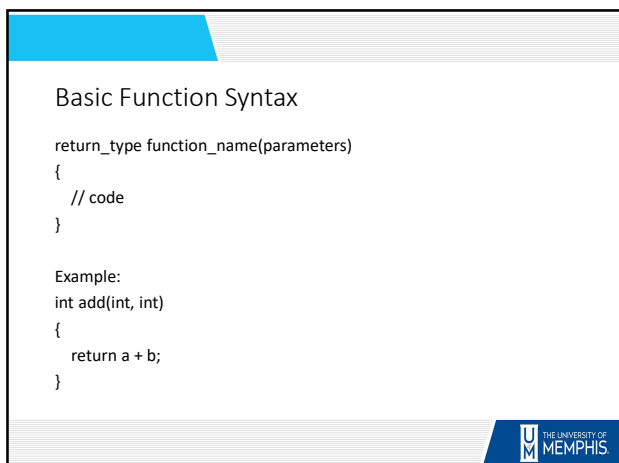




1



2



3

Function Prototypes

- Declares a function before it is used
- Tells compiler the function name, return type, and parameters

• Example:

```
int add(int a, int b);
```



4

Function Prototypes

- Note that both

```
int add(int a, int b);
```

and

```
int add(int, int);
```

Are valid for function prototypes.

Feature	Type Only (e.g., <code>int connect(char*, int);</code>)	Type + Name (e.g., <code>int connect(char *ip_address, int port);</code>)
Compiler Requirement	Fully sufficient.	Fully sufficient.
Code Readability	✗ Poor. It is unclear what the variables represent without checking the function definition.	✗ Excellent. Self-documenting; immediately shows what data is expected.
IDE Support	✗ Weak. Autocomplete/hover text will only show types, forcing you to guess the parameter order.	✗ Strong. Tooltips display the exact names, helping you pass arguments correctly.
Maintenance	✗ Slightly easier. Changing a parameter name in the <code>.c</code> file won't require updating the <code>.h</code> file.	✗ Slightly more work. If you rename variables, you should keep the prototype updated for clarity.



5

Why Use Function Prototypes?

- Allows functions to be used before definition
- Enables better program organization
- Required in multi-file embedded projects
- Prevents compiler errors



6

Prototype Example in Embedded C

```
#include <stdio.h>
int add(int a, int b); // prototype

int main()
{
    int result = add(2,3);
}

int add(int a, int b)
{
    return a + b;
}
```



7

Why Use Functions?

- Code reuse
- Easier debugging
- Improves structure
- Reduces duplication
- Essential for embedded applications



8

Example 1: Simple Function

```
int led_on()
{
    // turn LED on
    return 0;
}

int main()
{
    led_on();
}
```



9

Example 2: Function with Parameters

```
int multiply(int x, int y)
{
    return x * y;
}

int main()
{
    int result = multiply(3, 4);
}
```



10

Example 3: Function Called Multiple Times

```
int read_sensor()
{
    return 25; // mock value
}

int main()
{
    int t1 = read_sensor();
    int t2 = read_sensor();
    int t3 = read_sensor();
}
```



11

Example 4: Reusable Delay Function

```
void delay(int time)
{
    // loop for delay
}

int main()
{
    delay(1000);
    delay(1000);
    delay(1000);
}
```



12

Example 5: Function Calling Another Function

```
int square(int x)           int main()
{                           {
    return x * x;           int result = compute(5);
}                           }

int compute(int val)
{
    return square(val);
}
```



13

Embedded Systems Context

- Functions control hardware modules
- Used for drivers (UART, SPI, I2C)
- Improves maintainability
- Helps abstraction of hardware



14

Summary

- Functions are reusable blocks of code
- Prototypes define functions before use
- Functions can be called multiple times
- Functions can call other functions



15
