Function definition
- Gives a name to a group of statements which can then be executed (called) just using that name.
- Mathematical functions
  \[ \text{sin}(x) = y \]
  \[ ^{\text{^}} \quad ^{\text{^}} \quad ^\text{function's result} \]
  \[ ^{\text{^{}}+^{}} \quad \text{argument} \]
  \[ ^{\text{^{}}+-} \quad \text{function name} \]
- In C, function result is called return value.

Defining a function
- Functions must be defined to be used.
- Definition gives
  - type of its return value
  - function's name (same rules as variable names)
  - names and types of its parameters
  - its statements (or body)
  \[ \text{ftype} \text{fname} (\text{parameterlist}) \]
  \[ \{ \text{body of function} \} \]
- If type omitted an int type is assumed.
- If no return type is desired the term void should be used.
- If parameter list is empty the term void should be used.
- The body should have a return statement where the type matches the return type specified.
- The fname must be unique.
- A function can NOT be defined inside another one.
Simple example function definition:

```c
void error_msg(void) {
    printf("This ");
    printf("is Bad Input\n");
    return;
}
```

- function name = "error_msg"
- return type = nothing
- list of parameters = empty
- body has only three statements

Calling (executing) a function:

- general form:
  - `function-name(any arguments)`
- as a statement:
  - `printf("Hello,");`
- or as an expression in assignment:
  - `ch = scanf("%d%d%d",&a,&b,&c);`

Examples

- parameter passing example
- return value example
- function calling function example