CSE 220 – C Programming

C Fundamentals

Administration

- The first (real) Mimir Assignment will be emailed out on Thursday.
  - As always, the assignment will be due on the following Thursday at 10pm.
  - Be sure to click the button "I'm done with this test", else you won't receive credit.
  - The content will include material from Wednesday's and next Monday's lectures.

Outline

- Structure of a C program
- Functions
- Comments
- Variables
- Printing output
- Reading input
- Constants
- Identifiers
- Layout

http://www.imd.org/uupload/imd.website/wcc/Top_banners_fundamentals.jpg

Program Design

- The goal is to write a program
  - That has no compilation errors
  - Does what it is supposed to do (*no logical errors*)

- What are the steps to accomplish the goal?
- How should the steps be written?

Example

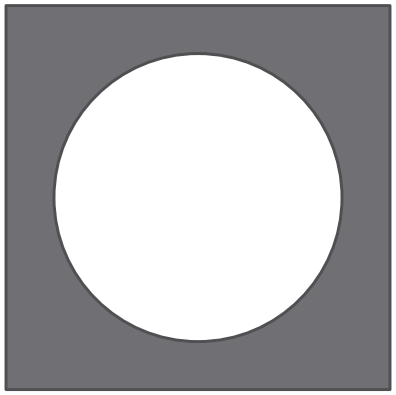
- Write a program that takes two numbers from the user and outputs their sum

Steps:

1. Ask the user for the first number
2. Record the first number
3. Ask the user for the second number
4. Record the second number
5. Compute the sum as 1st number + 2nd number
6. Record the result
7. Display the sum on the screen

Example

- Write a program that computes the area of the shaded section



Steps:

1. Ask the user for the square side
2. Record the value (call it s)
3. Ask the user for the circle radius
4. Record the value (call it r)
5. Compute the shaded area as $s^2 - 3.14*r^2$
6. Record the result
7. Display the output on the screen

First Program Explained

```
#include <stdio.h>
int main(void) {
  printf("Hello World!\n");
  return 0;
}
```

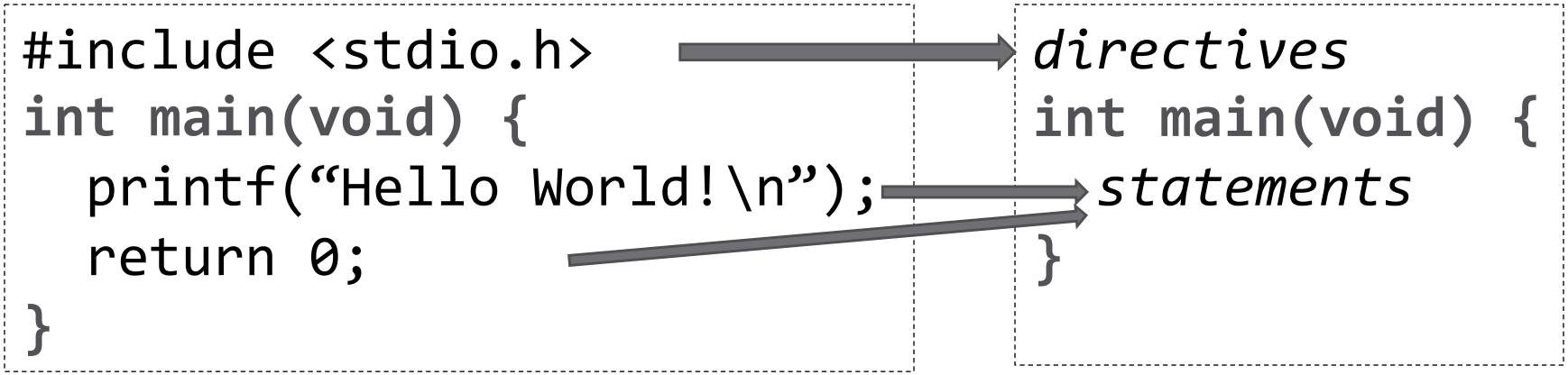
#include <stdio.h>: information in the header **stdio.h** needs to be included before the program is compiled

main: a function, the main position to start the program at

{}**:** delimit start and end of a function

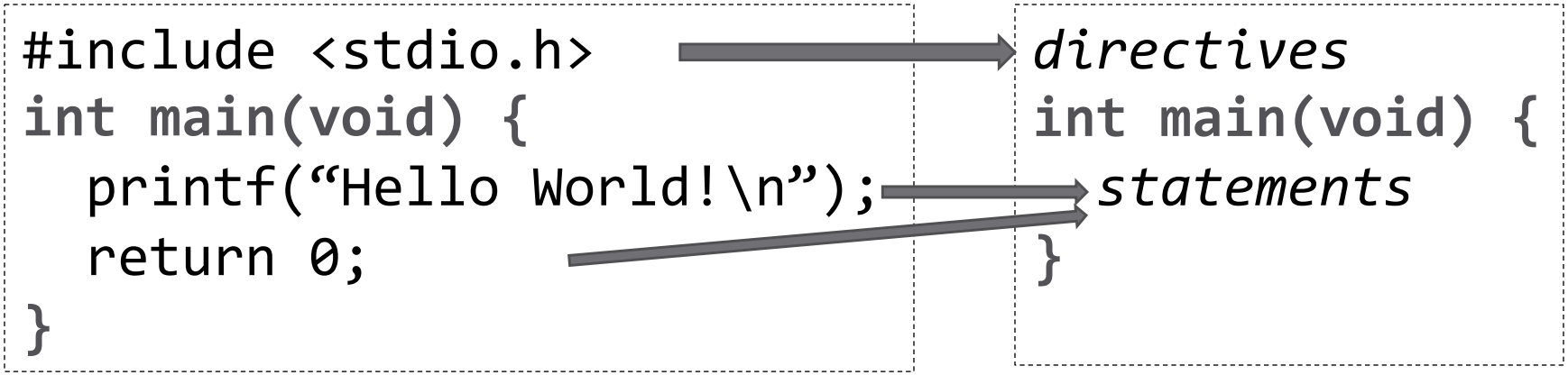
printf: a function, displays Hello World to the screen

Fundamentals of C



- directives:
  - commands for the preprocessor
  - one line long
  - begin with #
  - No semicolon at the end

Fundamentals of C



- statements:
  - commands to be executed when the program runs
  - End with semicolon (with some exceptions)

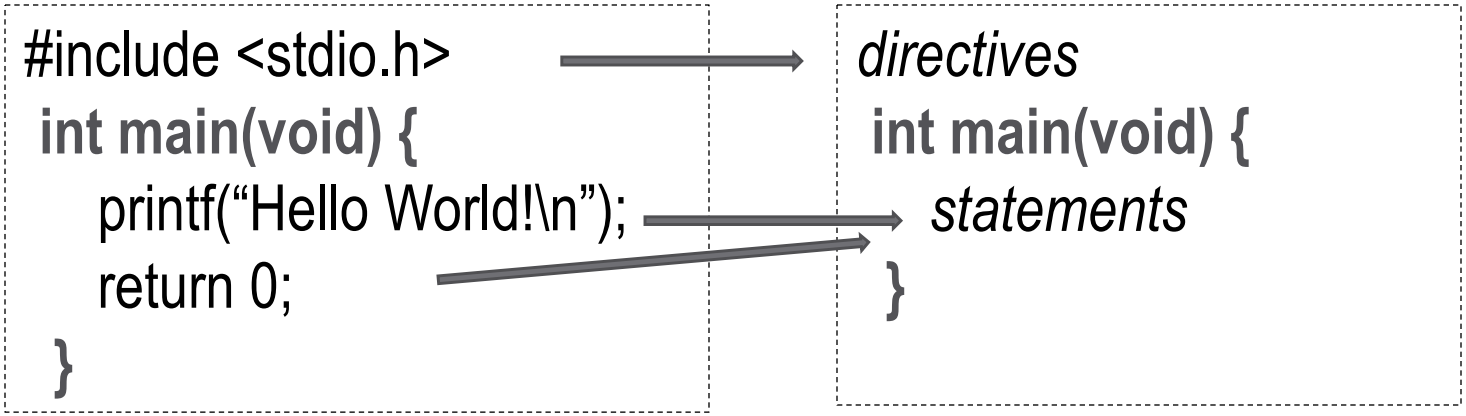
Examples

```
#include <stdlib.h>
#define RATE 0.8

…    …    ….

y = 2x + 5;
area = width * height;
guess = rand();
rand();
```

directives

statements

Fundamentals of C



The main function:

- main: function, returns a value
- int: return value is an integer
- void: main does not take any arguments
- return: terminates the function, returns a value
- printf: a call to function printf

Calling a function

- Functions have four parts:
  - A name (e.g. `printf`)
  - A body, which is a series of statements that run when a function is called
  - An argument list that gives information to a function
  - A return type, the one value a function can return.
- However, to call (use) a function, you just need to know its name and what arguments (parameters) it needs.
- `printf("Your age is %d", age);`

Name Arguments (2)

printf

- The `printf` function is used to output text to the screen.
- It takes one or more arguments:
  - The first argument is a character string (demarcated by quotation marks
    - For example: `"The price is %d"`
  - The following optional arguments are used to replace the placeholder characters (`%d`, `%f`, and others).
  - Don't forget the semicolon at the end of the statement.
  - `printf("The price for %d is %.2f", quantity, total);`
- `Special Characters`
  - `\n` (adds a newline character)
  - `%d` (placeholder for an integer)
  - `%f` (placeholder for a floating point number)
  - `%.3f` (placeholder for a float with three decimal places)

Examples

```
#include <stdio.h>
int main(void) {
  printf("Ready Set Go!\n");
  return 0;
}
```

Ready Set Go!

```
#include <stdio.h>
int main(void) {
  printf("Ready\n")
  printf("Set\n Go!\n");
  return 0;
}
```

Ready
Set
Go!

Comments

```
/*  HelloWorld.c
    Purpose: prints greeting */
#include <stdio.h>
int main(void) {
  /* Greet twice */
  printf("Hello World!\n"); //1st
  printf("Hello World!\n"); //2nd
  return 0;
}
```

- Provide documentation
- Ignored by the compiler
- May appear anywhere
- May extend over multiple lines (/* */)
- Cannot be nested
- // comments end at the end of the line

Comments

```
1:  /*
**************************** /
2:   *  HelloWorld.c
3:   *  Purpose: prints greeting
4:   *
**************************** /
5:   #include <stdio.h>
6:   int main(void) {
7:     /* Greeting #1 */
8:     printf("Hello World!\n");
9:     /* Do not show 2nd greeting
10:    /* Greeting #2 */
11:    printf("Hello World!\n"); */
12:    return 0;
13:  }
```

- Where does the first comment end?
- Where does second comment end?
- Some editors use different colors for comments. Helps in tracking comment termination.

Comments

```
1:  /*
**************************** /
2:    *  HelloWorld.c
3:    *  Purpose: prints greeting
4:    *
**************************** /
5:    #include <stdio.h>
6:    int main(void) {
7:       /* Greeting #1 */
8:       printf("Hello World!\n");
9:       /* Do not show 2nd greeting
10:     /* Greeting #2 */
11:     printf("Hello World!\n"); */
12:     return 0;
13:   }
```

- Where does the first comment end?
- Where does second comment end?
- Some editors use different colors for comments. Help track comment termination

Syntax error

Where does the multiline comment end?

At the first /*

At the last /*

At the first */

At the last */

Example

Write a program that takes two numbers from the user and outputs their sum

```
#include <stdio.h>
int main(void) {
    //Ask the user for the first number
    printf("Enter the first number\n");

    //Read and record the number that user enters
    …    …    …
    //Ask for second number, read it and record it …

    //Exit the function returning an integer
    return 0;
 }
```