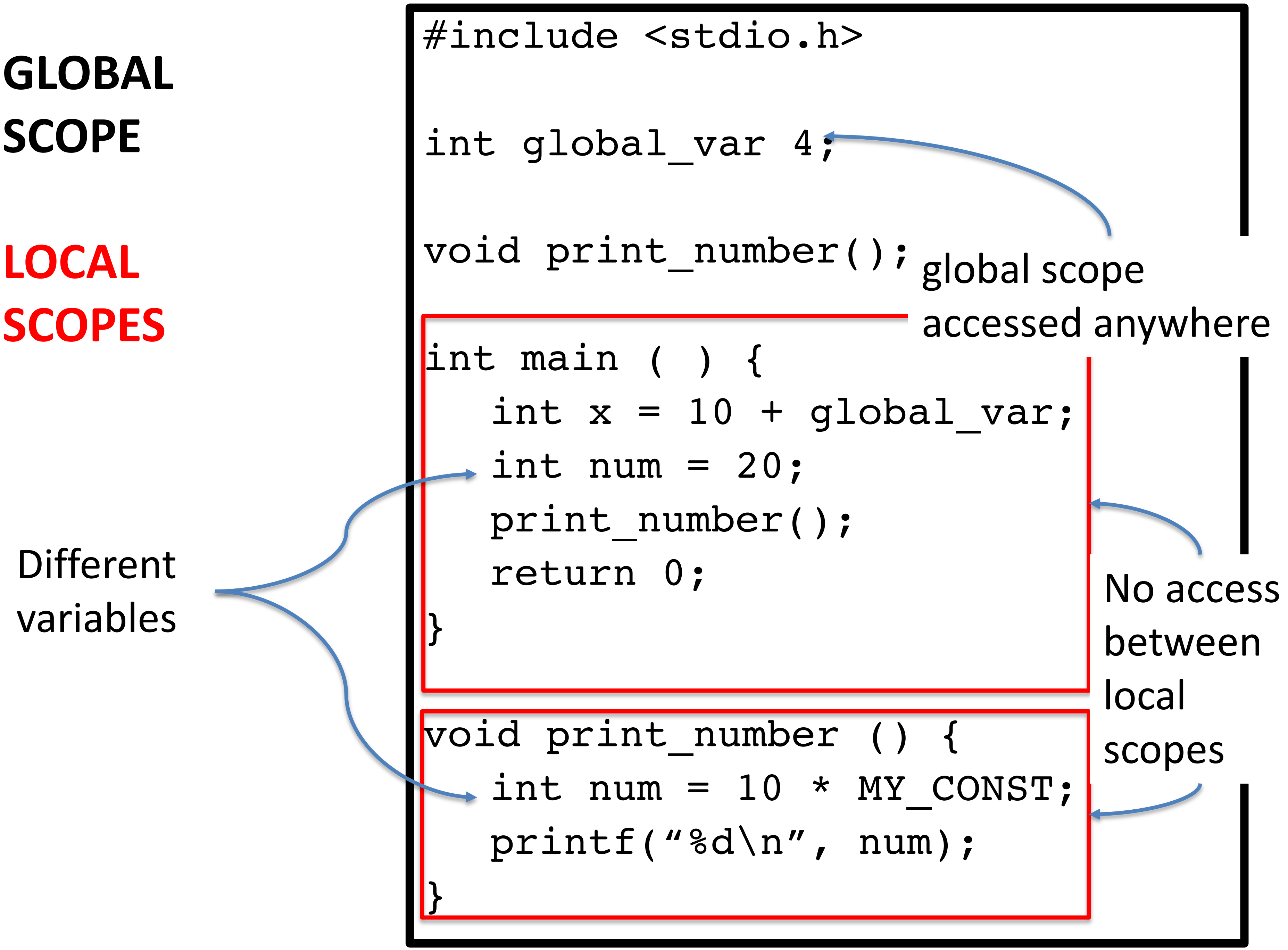
CSC 111 – Admin

- All classes, labs, etc. in person unless otherwise announced
- Waitlist – class has been expanded to max capacity
- Assignment 2 Posted – due Sunday.
- Lab marking happens in your registered lab

GLOBAL SCOPE

LOCAL SCOPES



```c
#include <stdio.h>

int global_var 4;

void print_number();

int main ( ) {
   int x = 10 + global_var;
   int num = 20;
   print_number();
   return 0;
}

void print_number () {
   int num = 10 * MY_CONST;
   printf("%d\n", num);
}
```

global scope accessed anywhere
No access between local scopes
Different variables

Defining a function that take arguements

General form:

Concrete example:

Function prototype

```c
void fn_name(type parameter_name);
```

```c
void print_number(int num);
```

Must match

```c
void fn_name (type parameter_name) {
    C statement 1;
    C statement 2;
}
```

```c
void print_number (int num) {
    printf("%d\n", num);
}
```

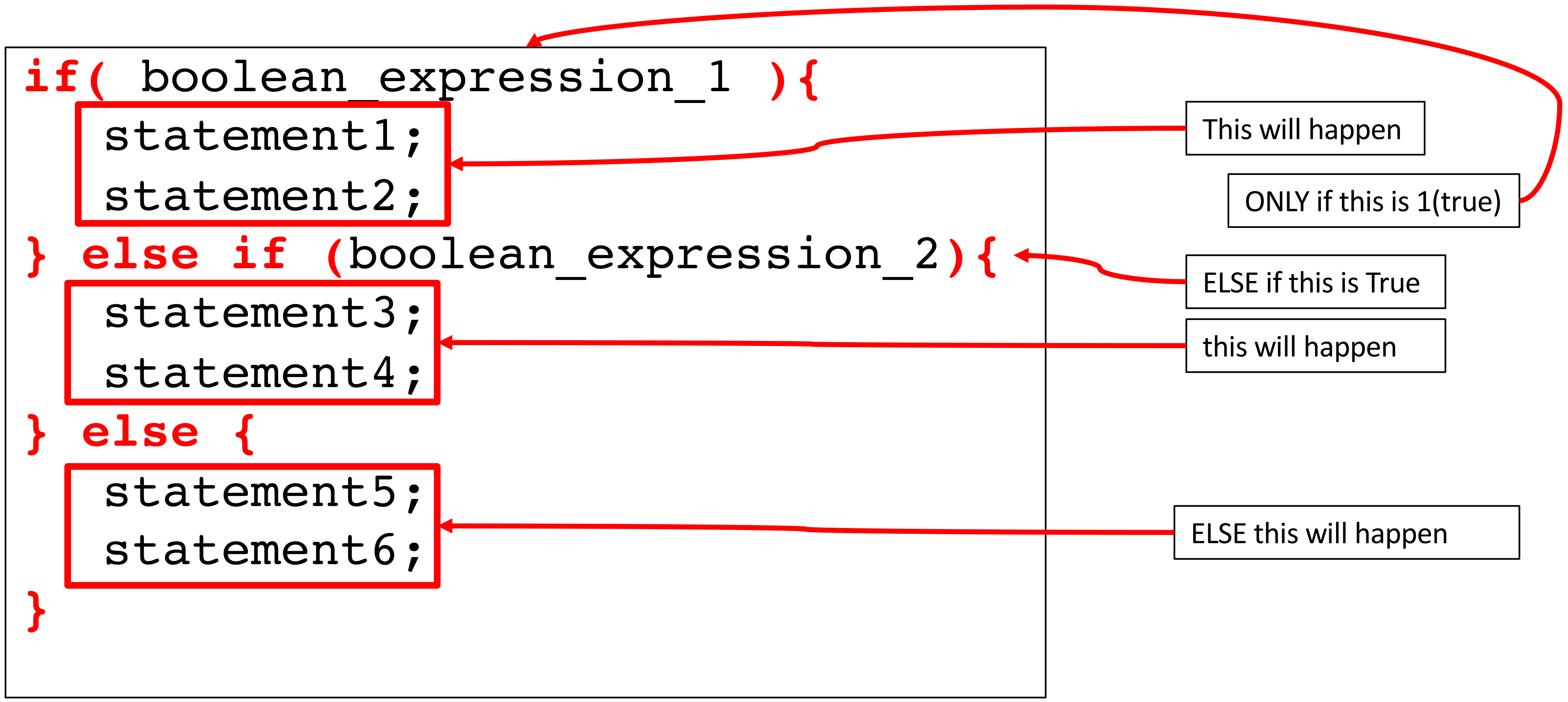
Function definition

Relational Operators and Boolean expressions

Relational Operator	meaning	Example Boolean expression	Result of Boolean expression
`>`	`greater than`	`x > y`	**1** `if x is greater than y,` **0** `otherwise`
`<`	`less than`	`x < y`	**1** `if x is less than y,` **0** `otherwise`
`>=`	`greater than or equal to`	`x >= y`	**1** `if x is greater than or equal to y,` **0** `otherwise`
`<=`	`less than or equal to`	`x <= y`	**1** `if x is less than or equal to y,` **0** `otherwise`
`==`	`equal to`	`x == y`	**1** `if x is equal to y,` **0** `otherwise`
`!=`	`not equal to`	`x != y`	**1** `if x is not equal to y,` **0** `otherwise`

False represented by 0
True represented by 1

Conditional statements – if/else if/else



```
if( boolean_expression_1 ){
    statement1;
    statement2;
} else if (boolean_expression_2){
    statement3;
    statement4;
} else {
    statement5;
    statement6;
}
```

GLOBAL SCOPE

```c
#include <stdio.h>

void print_number(int num);

int main ( ) {
    int x = 11;
    print_number(11);
    x = 10;
    print_number(10);
    return 0;
}
/* Purpose: ...
 *  Parameters: int num – a number
 */
void print_number (int num) {
    int x = 11;
    if(num == x) {
        int y = 2;
        num += (y + x);
        printf(“%d\n”, num);
    } else {
        printf(“%d\n”, num);
    }
    printf(“done\n”);
}
```

LOCAL SCOPES

Cannot access y
in these scopes

Tracing Scope – Example

- What is printed to the screen when the following function is called with the line: **foo(0);**
  - NOTE: documentation omitted intentionally.

```
void foo(int n) {
      int x = 12;
      if(n > 0) {
            int x = 14;
            x += 10;
      } else {
            int n = 20;
            x += 100;
      }

      printf("%d %d\n", x, n);
}
```

What abou foo(5); ?

Nested Conditions and Logical Operators

A conditional statements can be inside a conditional statement

```
if( boolean_expression_1 ){
   if( boolean_expression_2 ){
      statement1;
      statement2;
   } else {
      statement3;
      statement4;
   }


   ...
}
```

Logical operators…

Logical Operator	Example Boolean expression	Result of Boolean expression

`!`	`!(expr)`	`1 if expr is False,`
		`0 otherwise`

`&&`	`expr1 && expr2`	`1 if expr1 AND expr2 are True,`
		`False otherwise`

`||`	`expr1 ||  expr2`	`1 if expr1 OR  expr2 are True,`
		`0 otherwise`

RECALL: in C
False represented by 0
True represented by 1

Adding **logical operators** to precedence table

Precedence	Description	Associativity
Highest	Operations enclosed in brackets ( ), ++/–– postfix	left to right
	+/– unary operator, ++/–– prefix, (type) cast, !	right to left
	`*, /, %`	left to right
	`+, -`	left to right
	`<, <=, >, >=`	left to right
	`==, !=`	left to right
	`&&`	left to right
	`||`	left to right
Lowest	`=, +=, -=, *=, /=, %=`	right to left

Exception: Short circuit evaluation

Logical Operator	Example Boolean expression	Result of Boolean expression

`||`	`x < y || y < z`	`The x<y evaluates first ->`
		
		`If it is True,`
		`the rest of the expression is not evaluated`
		
		`If it is False, the y<z is evaluated,`
		`then finally the` **`||`** `expression is evaluated`

`&&`	`x < y && y < z`	`The x<y evaluates first ->`
		
		`If it is False,`
		`the rest of the expression is not evaluated`
		
		`If it is True, the y<z is evaluated,`
		`then finally the` **`&&`** `expression is evaluated`