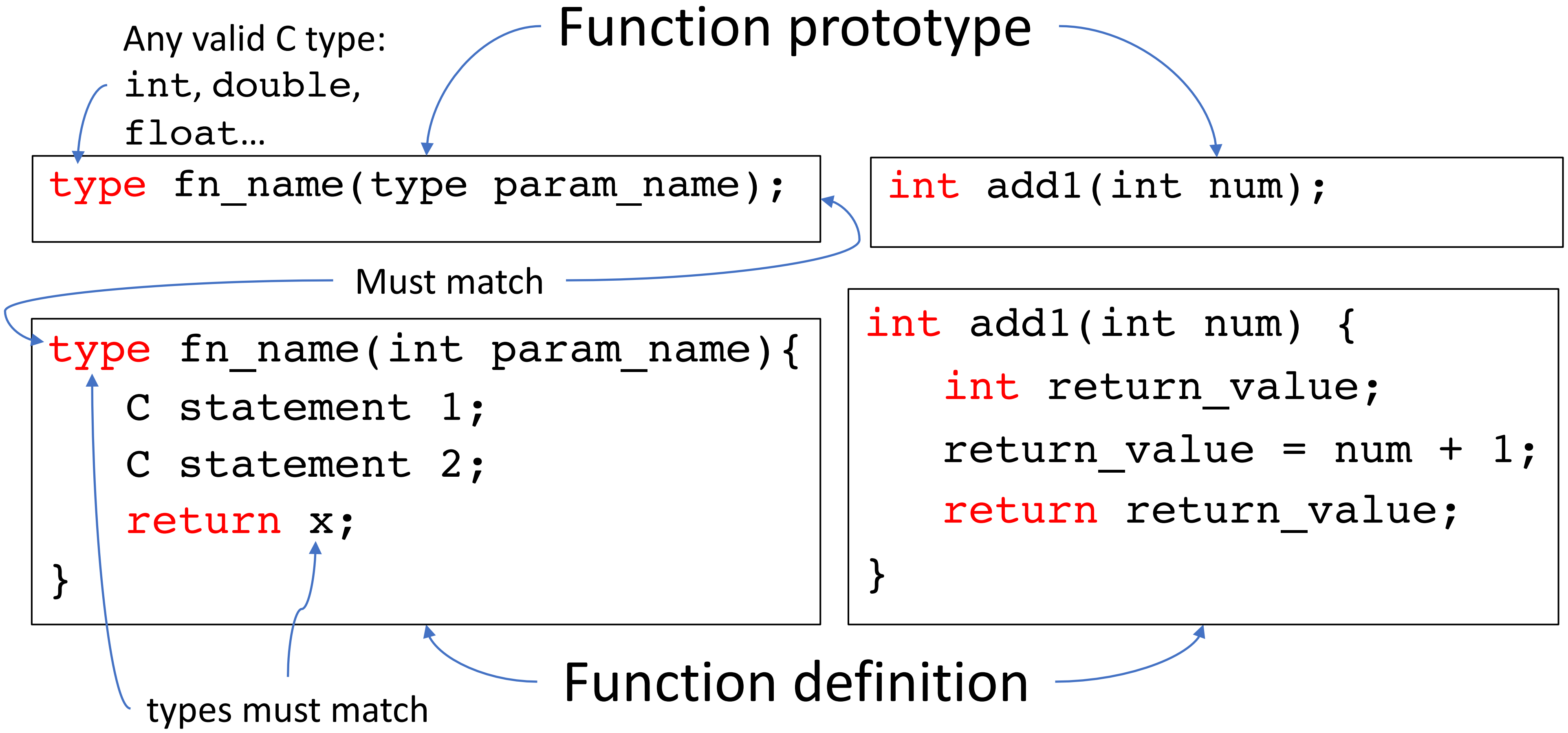
CSC111

Review - Defining a function with a return value

General form:

Concrete example:



Any valid C type: `int`, `double`, `float`…

Function prototype

```
type fn_name(type param_name);
```

```
int add1(int num);
```

Must match

```
type fn_name(int param_name){
   C statement 1;
   C statement 2;
   return x;
}
```

```
int add1(int num) {
   int return_value;
   return_value = num + 1;
   return return_value;
}
```

types must match

Function definition

```
#include <stdio.h>

int add1(int num);

int main ( ) {
        int x = 12;
        int result;
        result = add1(x);
        result = add1(11);
        return 0;
}

/* Purpose: calculates num + 1
 * Parameters: int num – a number
 * Returns: int – the value of num + 1
 */
int add1 (int num) {
        int return_value;
        return_value = num + 1;
        return return_value;
}
```

Function PROTOTYPE

Function CALLs
passing expected **argument**
storing the returned result

Documentation

Function DEFINITION

Count driven loops

To repeat something a specific number of times we must keep **count**!

Initial value of our counter

Condition asking if loop should continue

Increase our counter

```
int count = 0;
printf(“start:\n”);
while (count < 3) {
    printf(“%d\n”, count);
    count++;
}
printf(“%d: end\n”);
```

Code to execute if conditio is true (1)

The for loop…

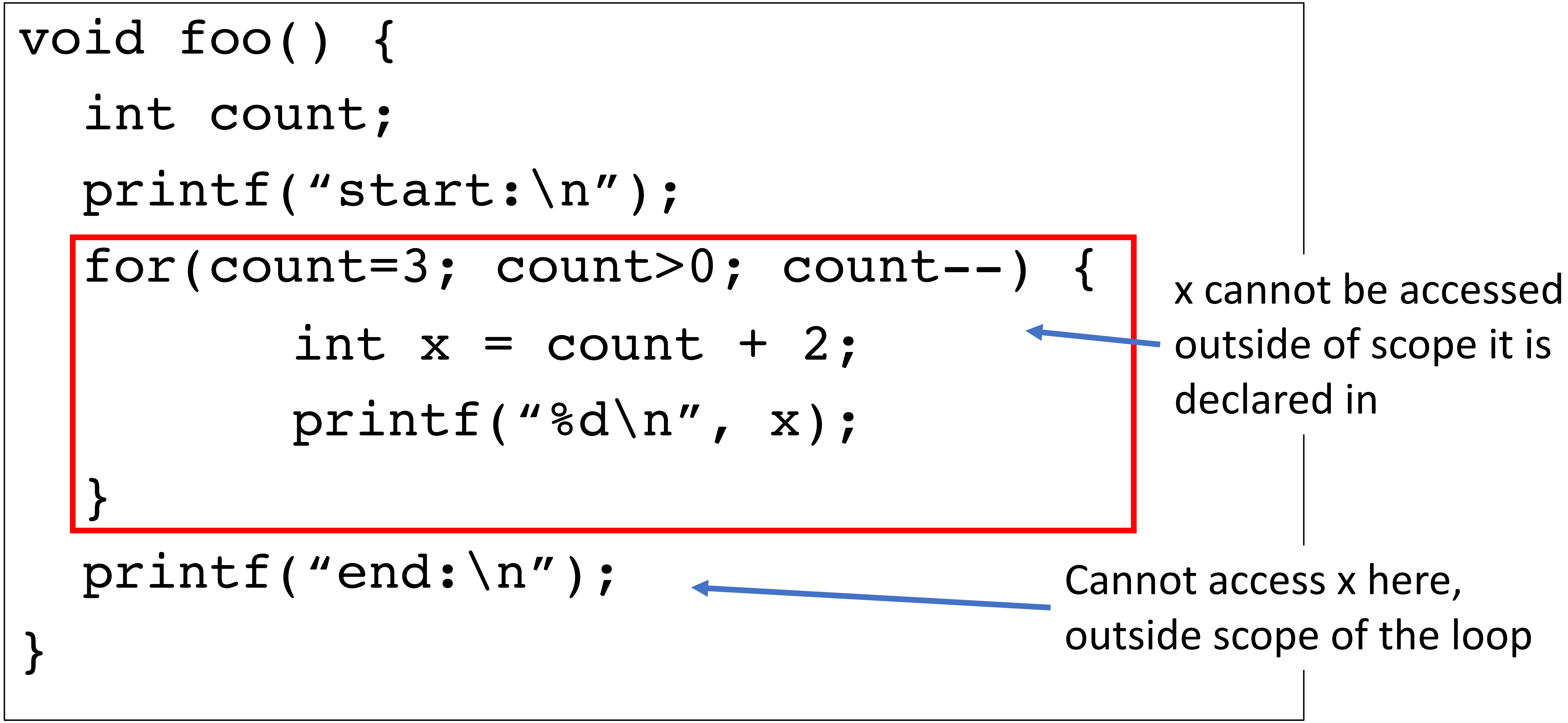
```
int count = 0;
printf(“start:\n”);
while (count<3) {
     printf(“%d\n”, count);
     count++;
}
printf(“%d: end\n”, count);
```

```
int count;
printf(“start:\n”);
for(count=0; count<3; count++) {
     printf(“%d\n”, count);

}
printf(“%d: end\n”, count);
```

Code to execute if condition is true (1)

Scope…



```
void foo() {
  int count;
  printf(“start:\n”);
  for(count=3; count>0; count--) {
        int x = count + 2;
        printf(“%d\n”, x);
  }
  printf(“end:\n”);
}
```

x cannot be accessed outside of scope it is declared in

Cannot access x here, outside scope of the loop

Demo

- Simple for loop -> while loop too.

Infinite Loops

- Condition is always true
  - The program will theoretically go forever
  - Stop infinite loop by pressing ‘control’ and ‘c’

Demo – Infinite Loop

Tracing loops

```
int main( void ) {
    int limit = 4;
    int counter;
    printf( "start: " );

    for (counter = 0; counter < limit; counter++) {
        printf( "%d ", counter);
    }
    printf( "end!\n" );

    return 0;
}
```

Demo – Aproximating Pi

Leibniz’s Formula

$$X = 4 - 4/3 + 4/5 - 4/7 + 4/9 - \ldots.$$

- We will create 2 variables sum, d (denominator)
- Initialize sum = 0
- Initialize d = 1
- Create a loop
- Loop to 1000000 (bigger number = more precision)
- Check if i is even then sum=sum+4/d, else sum=sum-4/d
- Increment d by 2 every at every iteration
- Print sum