Using Functions

Prime Testing

```
int isPrime(int n) {
    int divisor;

    if (n == 2)
        return 1; // 2 is not a prime

    if (n == 1 || n % 2 == 0)
        return 0; // 1 is non-prime, so are even numbers

    for (divisor = 3; divisor * divisor <= n; divisor += 2)
        if (n % divisor == 0)  // non-prime
            return 0;

    return 1;
}
```

Using Functions

Prime Testing

```
int main() {
    int n;

    printf("Enter a number: ");
    scanf("%d", &n);

    if (isPrime(n))
        printf("%d is prime\n", n);
    else
        printf("%d is not prime\n", n)
}
```

Using Functions

GCD

Property

If m, n, q, r are integers and $m = nq + r$ then $(m, n) = (n, r)$

Proof.

- GCD of $m = nq + r, n$ should also be divisor of r.
- So it must also be a greatest common division or n, and r.
- I.e., GCD(m, n) = GCD(n, r).

□

Using Functions

GCD (contd)

Theorem

Let m and n be +ve integers. Then there is an algorithm that finds (m, n).

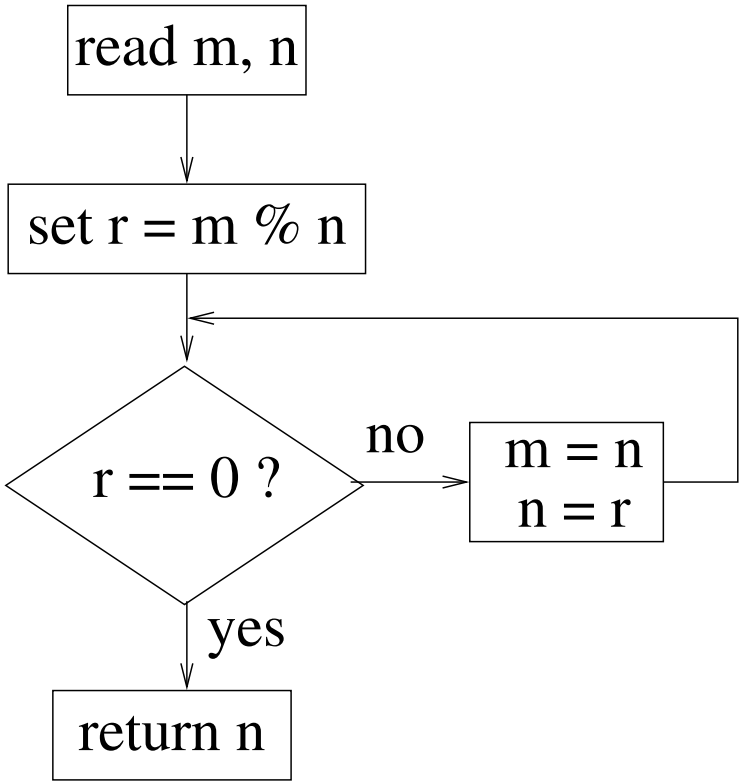
Proof.

$$\begin{aligned} m &= nq_1 + r_1, & (m, n) &= (n, r_1) \\ n &= r_1 q_2 + r_2, & (n, r_1) &= (r_1, r_2) \\ r_1 &= r_2 q_3 + r_3, & (r_1, r_2) &= (r_2, r_3) \\ r_2 &= r_3 q_4 + r_4, & (r_2, r_3) &= (r_3, r_4) \\ & \ldots \end{aligned}$$

Therefore, $\text{GCD}(m, n) = \text{GCD}(n, r_1) = \ldots = \text{GCD}(r_{n-1}, r_n) = r_n$. □

Using Functions

GCD (contd)



Using Functions

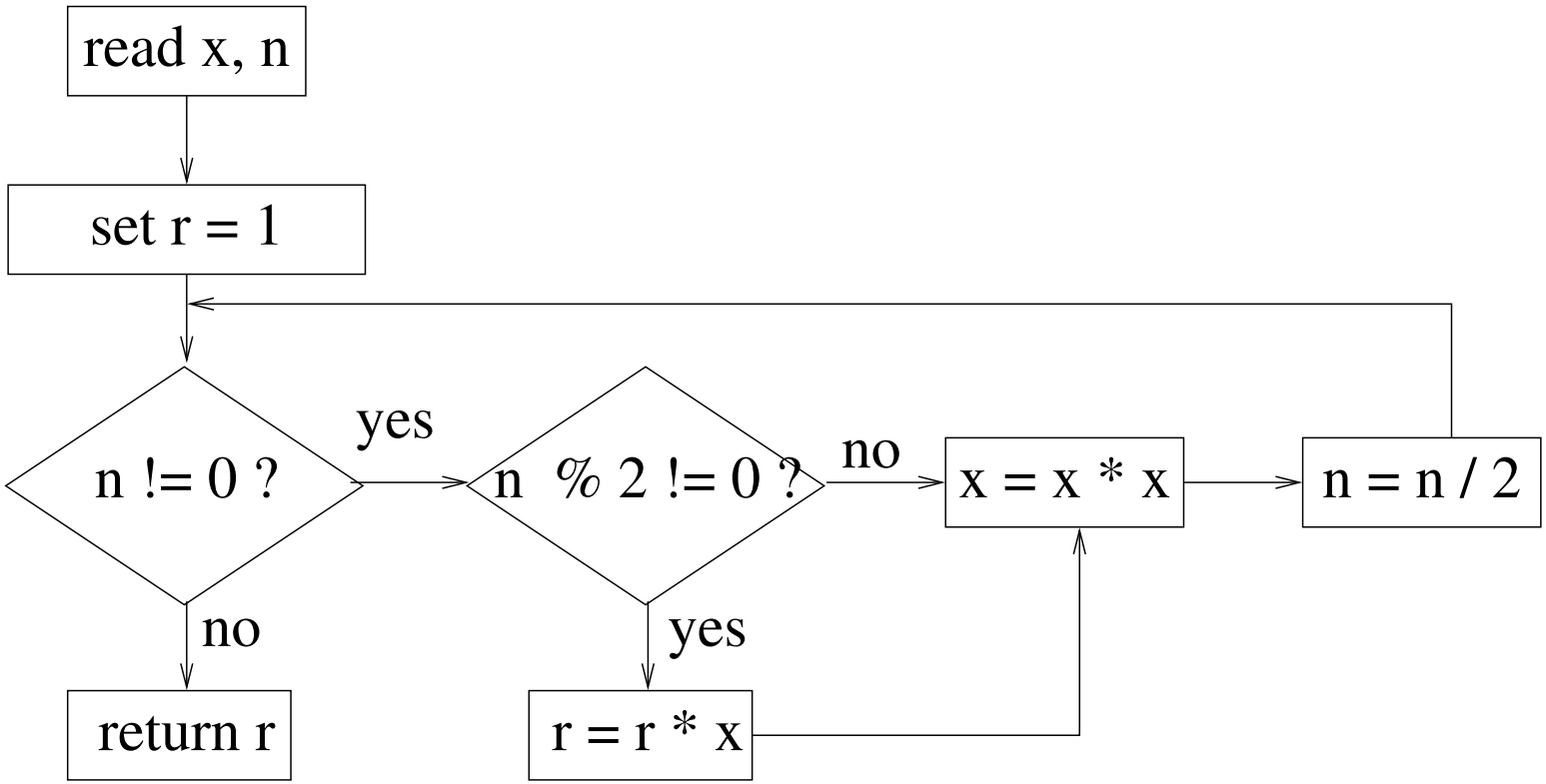
GCD (contd)

```
#include <stdio.h>
int gcd(int m, int n) {
    int r;
    if (n > m) {
        r = m; m = n; n = r;
    }
    r = m % n;
    while (r != 0) {
       m = n;   n = r;   r = n % m;
    }
    return n;
}

int main() {
    int m, n;
    printf("Enter m and n: ");
    scanf("%d %d", &m, &n);
    printf("gcd(%d, %d) = %d \n", n, m, gcd(n,m));
}
```

Using Functions

More Example (x power n)



Using Functions

More Example (x power n)

```
#include <stdio.h>
int XpowerN(int x, int n) {
    int r = 1;

    while (n != 0) {
        if (n % 2 != 0)
            r = r * x;
        x = x * x;
        n = n / 2;
    }
    return r;
}
/*** Main function appears in next slide ***/
```

Using Functions

More Example (x power n)

```
int main() {
    int x, n;

    printf("Enter x and n: ");
    scanf("%d %d", &x, &n);

    printf("%d raised power %d = %d\n", x, n, XpowerN(x, n));
}
```

Variable Scope

Global Variable

- Functions can communicate through `global` variables.
- Global variables are declared outside functions. and have **file scope**.
- Visible to all functions appearing after declaration.
- So, scopes can be restricted by placing declarations at different points.
- Have static storage allocation.

Variable Scope

Local Variable

- A variable declared inside a function or a block has **block scope**.
- Scope terminates within the block where it is declared.
- But declaration of the variable with same name in an inner block over-rides outer block scope.
- Have automatic storage duration
  - storage automatically allocated when block is executed.
  - automatically deallocated when block is exited.

Variable Scope

Static Local Variables

- By default, local variables have automatic storage duration
- Putting word `static` causes static instead of automatic allocation.
- Static storage retains the allocation throughout the program execution.
- Like automatic variables, static variables have block scope, therefore, not visible outside the block.

Parameters

- Have same properties as local variable:
  - Automatic storage duration
  - Block scope