C-Practical — Recursive Programs

Factorial • Fibonacci • Power Function

What is Recursion?

- A function calling itself to solve a problem
- Problem is divided into smaller sub-problems
- Must contain a Base Case to stop recursion
- Used in mathematical and divide-and-conquer problems

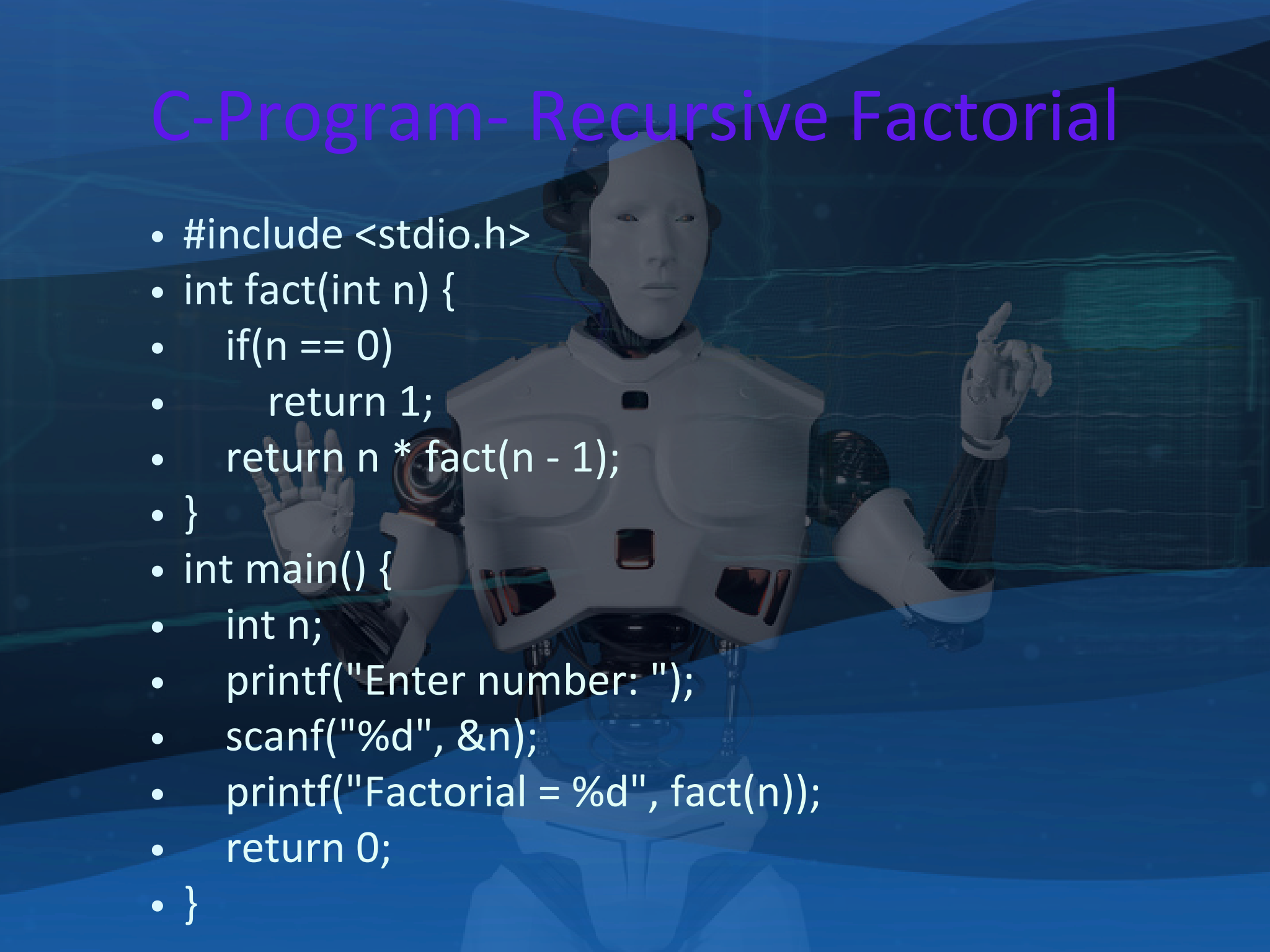
Structure of a Recursive Function

- Base Case — Terminates recursion
- Recursive Case — Function calls itself
- Problem size must reduce in every call
- Without base case → infinite recursion

Recursive Factorial -Concept

- Factorial (n!) = n × (n-1) × (n-2) … × 1
- Example: 5! = 120
- Base Case: 0! = 1
- Recursive Case: n! = n × fact(n-1)

C-Program- Recursive Factorial



```
#include <stdio.h>
int fact(int n) {
    if(n == 0)
        return 1;
    return n * fact(n - 1);
}
int main() {
    int n;
    printf("Enter number: ");
    scanf("%d", &n);
    printf("Factorial = %d", fact(n));
    return 0;
}
```

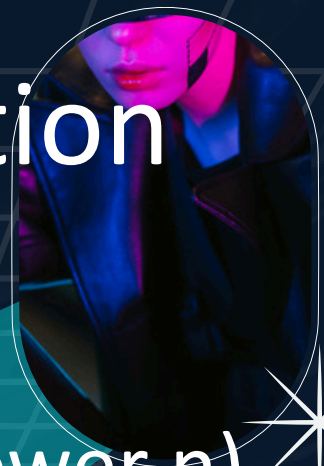
Recursion Fibonacci-Concept

- Fibonacci Series: 0, 1, 1, 2, 3, 5, 8…
- Each term = sum of previous two terms
- Base Cases: F(0)=0 , F(1)=1
- Recursive Case: F(n)=F(n-1)+F(n-2)

C Program- Recursive Fibonacci

```
#include <stdio.h>
int fib(int n) {
    if(n == 0) return 0;
    if(n == 1) return 1;
    return fib(n-1) + fib(n-2);
}
int main() {
    int n, i;
    printf("Enter terms: ");
    scanf("%d", &n);
    for(i = 0; i < n; i++)
        printf("%d ", fib(i));
    return 0;
}
```

Recursive Power Function



- Computes a^n (a raised to power n)
- Example: 2^4 = 16
- Base Case: a^0 = 1
- Recursive Case: a^n = a × a^(n-1)

C-Program Recursive Power Function

```
#include <stdio.h>
int power(int a, int n) {
    if(n == 0)
        return 1;
    return a * power(a, n - 1);
}
int main() {
    int a, n;
    printf("Enter base and power: ");
    scanf("%d%d", &a, &n);
    printf("Result = %d", power(a, n));
    return 0;
}
```

Advantages of Recursion

- Simple and clean logic
- Matches mathematical formulas
- Useful in tree & graph problems
- Helpful in divide-and-conquer algorithms

Limitation of Recursion

- Slower than iteration in some cases
- Consumes more memory (stack frames)
- May cause stack overflow for large input
- Requires carefully defined base case

Thank You