Shri Vile Parle Kelavani Mandal's

SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

Computer Engineering Department

Unit 4 -Function

Course: Programming in C

Course Code: PRC238912

Name of Staff: Mr. Pratik H. Shah

SEMESTER : II

DIVISION : A

Course Outcome – 3

Implement modular approach in programming

Student will be able to

- Implement modular approach in programming
- Develop user defined function for real time application

Introduction to Function

1. A function is a block of code that performs a specific task.
2. we can divide a large program into the basic building blocks known as function.
3. A function can be called multiple times to provide reusability and modularity to the C program.

Advantages

1. we can avoid rewriting same logic/code again and again in a program.
2. We can call C functions any number of times in a program and from any place in a program.
3. We can track a large C program easily when it is divided into multiple functions.
4. Reusability is the main achievement of C functions.

Types of function

There are two types of function in C programming:

1. User-defined functions
2. Standard library functions

User-defined function

You can also create functions as per your need. Such functions created by the user are known as user-defined functions.

Standard library functions

1. Built-in functions in C programming.
2. These functions are defined in header files. The printf() is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in the stdio.h header file.
3. To use the printf()function, we need to include the stdio.h header file using #include <stdio.h>.
4. The sqrt() function calculates the square root of a number. The function is defined in the math.h header file.

ELEMENTS OF USER-DEFINED FUNCTION

1. Function Declaration
2. Function Definition
3. Function Call

Function Declaration

1. It is also known as function prototype
2. It specifies function's name, parameters and return type.
3. It doesn't contain function body.
4. A function prototype gives information to the compiler that the function may later be used in the program.

Syntax of function prototype

```
returnType functionName(type1 parameter1, type2 parameter2, ...);
```

Function Definition

1. Function definition contains the block of code to perform a specific task.

Syntax of function definition

```
returnType functionName(type1 parameter1, type2 parameter2, ...)
{
   //body of the function
}
```

Function Call

1. Control of the program is transferred to the user-defined function by calling it.

Syntax of function call

```
functionName(argument1, argument2, ...);
```

Example: User-defined function

```
#include <stdio.h>
void addNumbers();                    // function declaration
void addNumbers()                     // function definition
{
   int result,a,b;
    a = 5;
    b = 10;
   result = a+b;
   printf("result=%d",result);
}

void main()
{
    addNumbers();                     // function call
}
```

What is Parameter ?

- In C Programming Function Passing Parameter is Optional.
- We can Call Function Without Passing Parameter .
- **Function With Parameter :**
  - add(a,b);
- Here Function add() is Called and 2 Parameters are Passed to Function.
- a,b are two Parameters.
- **Function Call Without Passing Parameter :**
  - Display();

1. Parameter : The names given in the function definition are called Parameters.
   - Formal Parameter :
     Parameter Written In Function Definition is Called "Formal Parameter".
   - Actual Parameter :
     Parameter Written In Function Call is Called "Actual Parameter".

2. Argument : The values supplied in the function call are called Arguments.

```
void main()
{
int num1;
display(num1);
}

void display(int para1)
{
-----------
-----------
}
```

Para1 is “Formal Parameter“

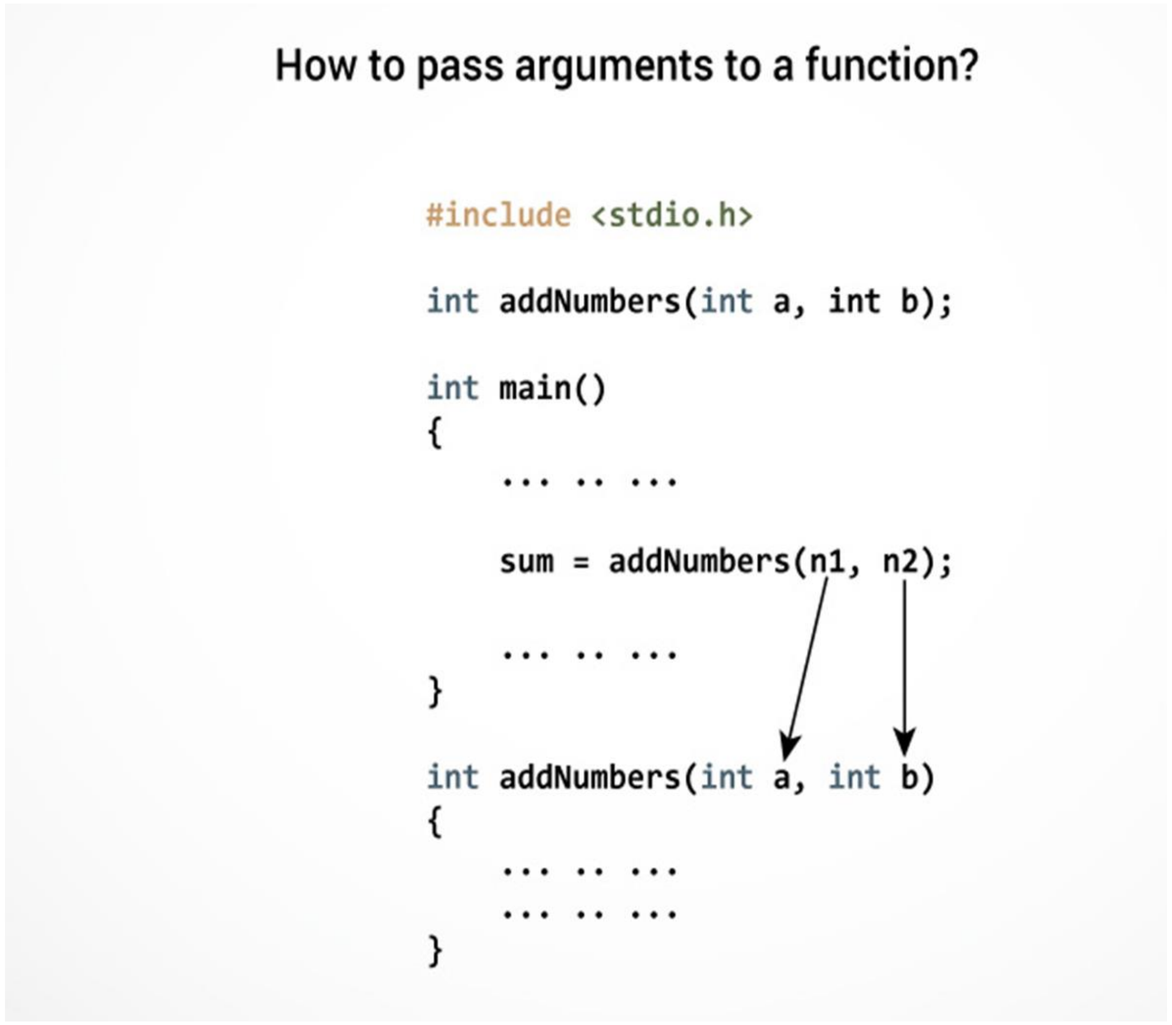
```
void main()
{
int num1;
display(num1);
}

void display(int para1)
{
-----------
-----------
}
```

num1 is “Actual Parameter“

Passing arguments to a function

In programming, argument refers to the variable passed to the function.



How to pass arguments to a function?

```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    ... .. ...
}
```

Return Statement

1. The return statement terminates the execution of a function and returns a value to the calling function.

2. The program control is transferred to the calling function after return statement.

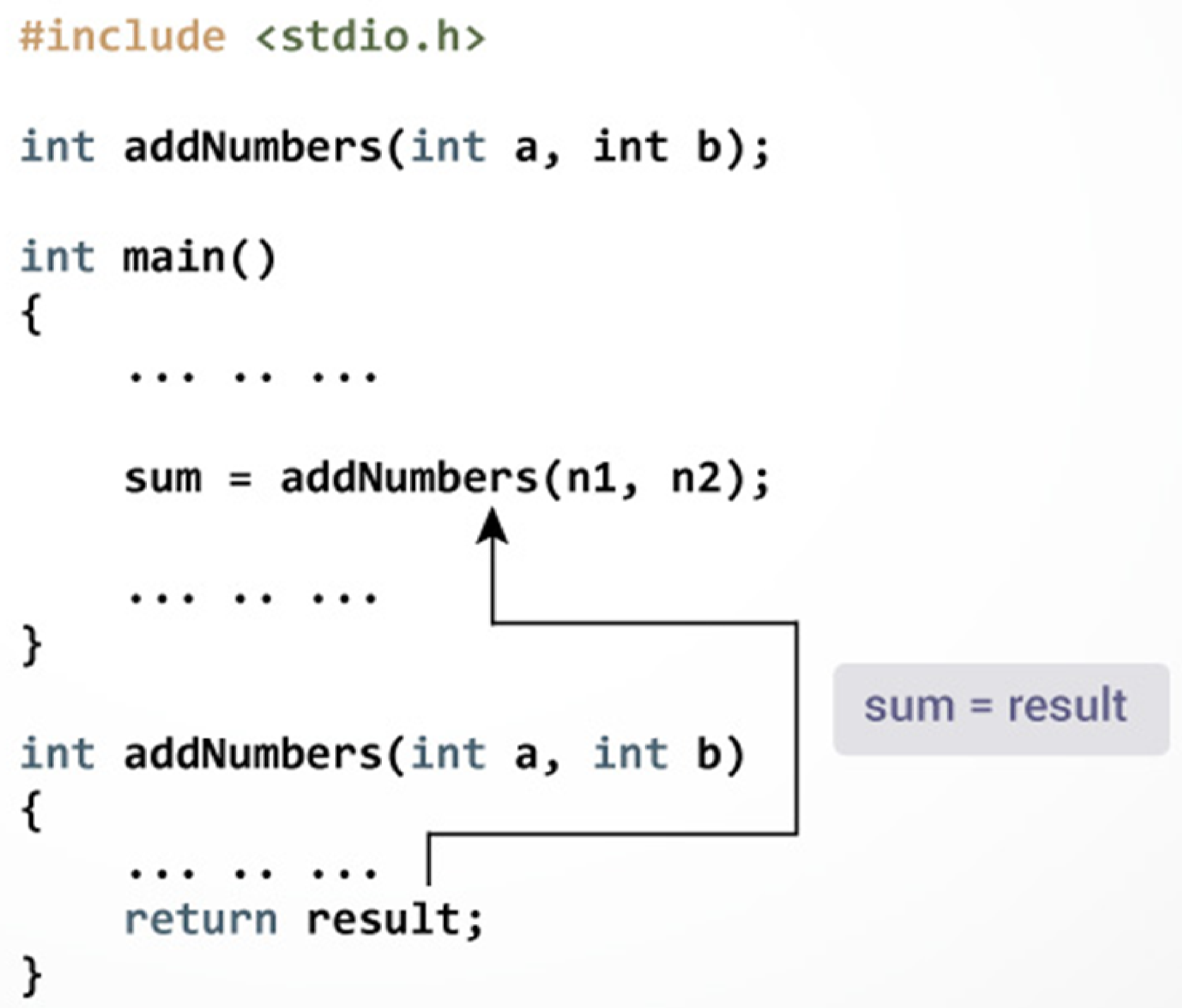
Syntax of return statement

```
return (expression);
```

Example

```
return a;
return (a+b);
```

Return statement of a Function



```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    return result;
}
```

Example #1:
No arguments passed and no return Value

```
#include <stdio.h>
void prime();
void prime()
{
   int n, i, flag=0;
   printf("Enter a positive integer: ");
   scanf("%d",&n);
   for(i=2; i <= n/2; i++)
   {
      if(n%i == 0)
      {
         flag = 1;
      }
   }
   if (flag == 1)
      printf("%d is not a prime number.", n);
   else
      printf("%d is a prime number.", n);
}
void main()
{
   prime();
}
```

Example #2:
No arguments passed but a return value

```
#include <stdio.h>
int prime();
int prime()
{
   int n;
   printf("Enter a positive integer: ");
   scanf("%d",&n);
   return n;
}
void main()
{
   int n, i, flag = 0;
    n = prime();
   for(i=2; i<=n/2;i++)
   {
      if(n%i==0){
         flag = 1;
         break;
      }
   }
   if (flag == 1)
      printf("%d is not a prime number.", n);
   else
      printf("%d is a prime number.", n);
}
```

Example #3: Argument passed but no return value

```
#include <stdio.h>
void prime(int n);
void prime(int n)
{
   int i, flag = 0;
   for(i=2; i <= n/2; ++i)
   {
      if(n%i == 0){
         flag = 1;
         break;
      }
   }
   if(flag == 1)
      printf("%d is not a prime number.",n);
   else
      printf("%d is a prime number.", n);
}
void main()
{
   int n;
   printf("Enter a positive integer: ");
   scanf("%d",&n);
   prime(n);
}
```

Example #4: Argument passed and a return value

```
#include <stdio.h>
int prime(int n);
int prime(int n)
{
    int i;
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0)
            return 1;
    }
    return 0;
}
void main()
{
    int n, flag;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    flag = prime(n);
    if(flag==1)
        printf("%d is not a prime number",n);
    else
        printf("%d is a prime number",n);
}
```

Which approach is better?

1. Well, it depends on the problem you are trying to solve.

2. In case of this problem, the last approach is better.

3. A function should perform a specific task.

4. The prime() function doesn't take input from the user nor it displays the appropriate message.

5. It only checks whether a number is prime or not, which makes code modular, easy to understand and debug.

call by value

1. The **call by value** method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function.
2. In this case, changes made to the parameter inside the function have no effect on the argument.
3. By default, C programming uses *call by value* to pass arguments.
4. In general, it means the code within a function cannot alter the arguments used to call the function.

```
#include <stdio.h>
void swap(int x, int y);
void swap(int x, int y)
{
   int temp;
   temp = x;
   x = y;
   y = temp;

}
void main ()
{
int a = 100;
int b = 200;
   printf("Before swap, value of a : %d\n", a );
   printf("Before swap, value of b : %d\n", b );
   swap(a, b);
   printf("After swap, value of a : %d\n", a );
   printf("After swap, value of b : %d\n", b );
}
```

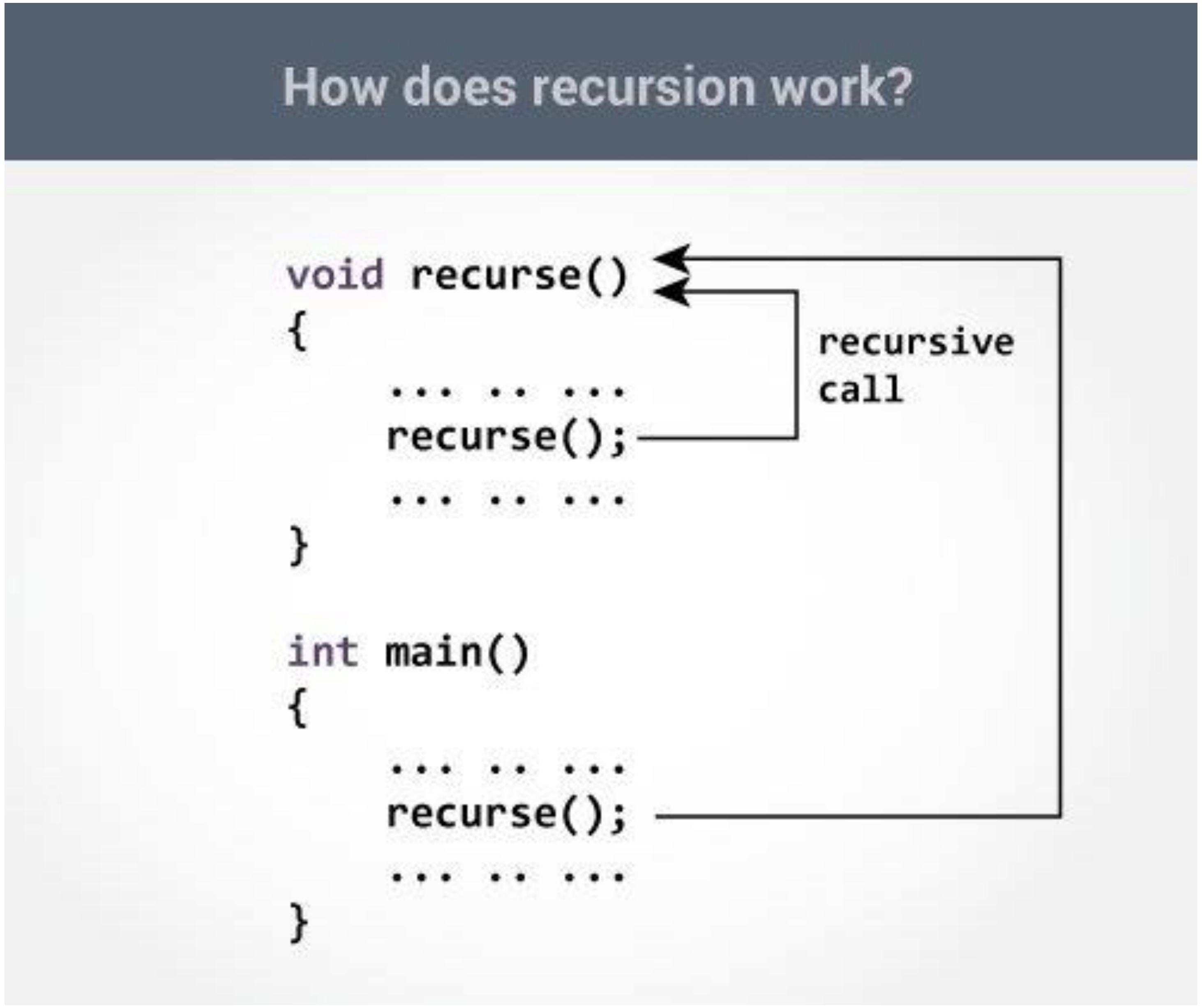
call by reference

1. The **call by reference** method of passing arguments to a function copies the address of an argument into the formal parameter.
2. Inside the function, the address is used to access the actual argument used in the call.
3. It means the changes made to the parameter affect the passed argument.
4. To pass a value by reference, argument pointers are passed to the functions just like any other value.
5. So accordingly you need to declare the function parameters as pointer types as in the following function **swap()**, which exchanges the values of the two integer variables pointed to, by their arguments.

```
#include <stdio.h>
void swap(int *x, int *y);
void swap(int *x, int *y)
{
   int temp;
   temp = *x;
   *x = *y;
  *y = temp;
}
void main ()
{
int a = 100;
int b = 200;
   printf("Before swap, value of a : %d\n", a );
   printf("Before swap, value of b : %d\n", b );
    swap(&a, &b);
   printf("After swap, value of a : %d\n", a );
   printf("After swap, value of b : %d\n", b );
 }
```

Recursion Function

1. A function that calls itself is known as a recursive function. And, this technique is known as recursion.
2. Recursion is the process of repeating items in a self-similar way.
3. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.
4. The C programming language supports recursion, i.e., a function to call itself.
5. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.
6. Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.



How does recursion work?

```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```

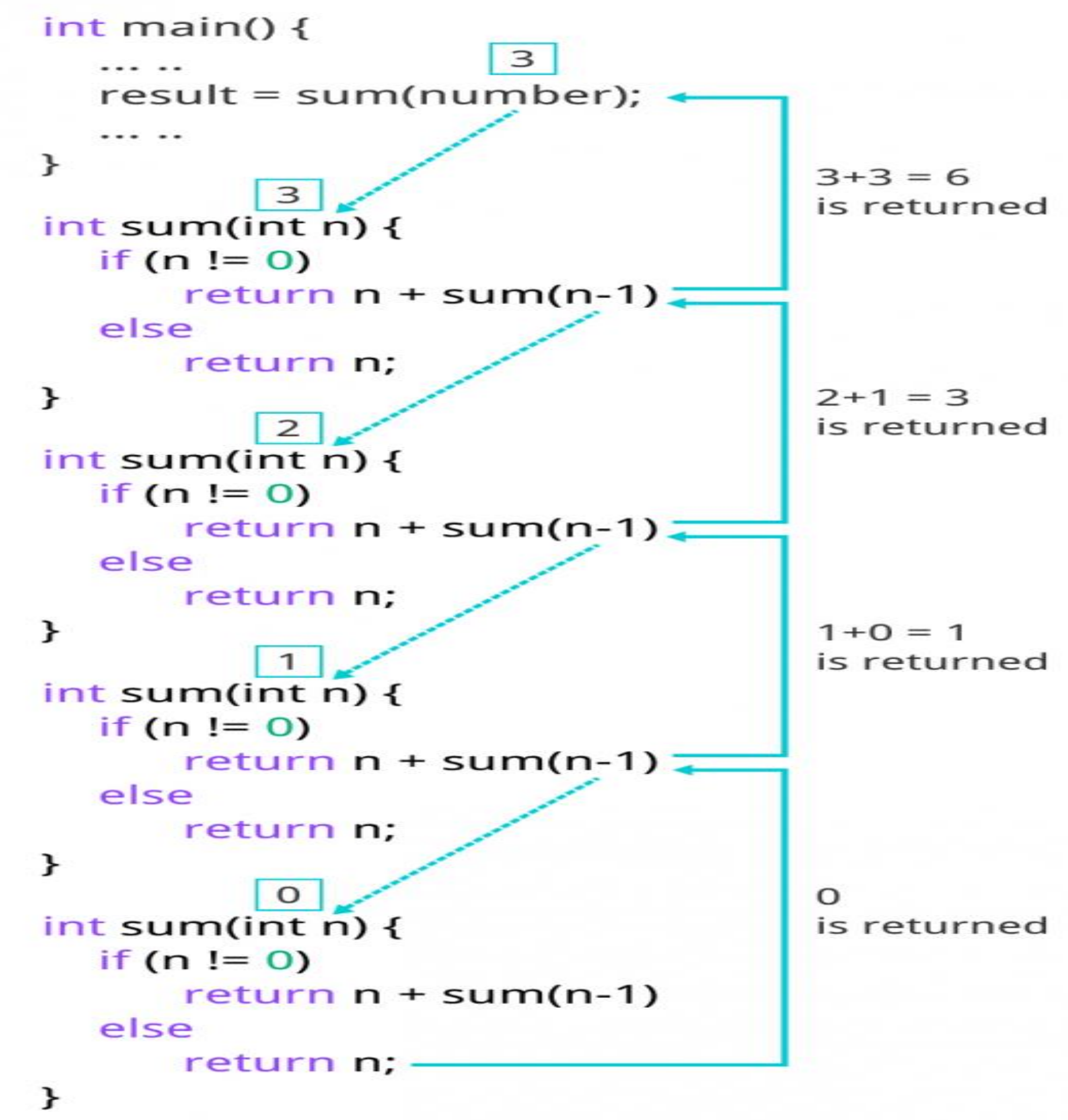
Example: Sum of Natural Numbers Using Recursion

```
#include <stdio.h>
int sum(int num);
int sum(int num)
{
   if (num!=0)
      return num + sum(num-1); // sum() function calls itself
   else
      return num;
}
void main()
{
   int number, result;

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

   result = sum(number);

   printf("sum=%d", result);
}
```



```
int main() {
   ... ..
   result = sum(number);
   ... ..
}

int sum(int n) {
   if (n != 0)
        return n + sum(n-1)
   else
        return n;
}

int sum(int n) {
   if (n != 0)
        return n + sum(n-1)
   else
        return n;
}

int sum(int n) {
   if (n != 0)
        return n + sum(n-1)
   else
        return n;
}

int sum(int n) {
   if (n != 0)
        return n + sum(n-1)
   else
        return n;
}
```

Advantages of Recursion

1. Recursion makes program elegant and cleaner.
2. All algorithms can be defined recursively which makes it easier to visualize and prove.

Disadvantages of Recursion

1. If the speed of the program is vital then, you should avoid using recursion.
2. Recursions use more memory and are generally slow. Instead of that, you can use loop

Exercise

1. Write a program in C to check if a given number is even or odd using the function.
2. Write a program in C to swap two numbers using a function**(Call by value and call by reference method**).
3. Write a program in C to check whether a number is a prime number or not using the function.
4. Write a program in C to calculate the sum of numbers from 1 to n using recursion.
5. Write a program in C to print the Fibonacci Series using recursion.
6. Write a program in C to find the Factorial of a number using recursion.
7. Write a program in C to print the first 50 natural numbers using recursion
8. Write a program in C to search a number from a list using the function.

Theory Questions

1. Explain Elements of User defined Function with example
2. Define Recursion
3. Explain different categories of function with example programs
4. Difference between call by value and call by reference
5. Define following terms: a) Parameter b) Argument
6. State Advantages of function
7. State advantages and disadvantages of Recursion function

THANK YOU !!!!