C

FUNCTIONS

✓c functions are classified as **user defined** and **library** functions

✓example for library functions are **printf**,**scanf sqrt**,**strcat** etc.

✓main() is an example for user defined function. Every program must have a main

✓library functions are pre-defined

✓user defined functions should be developed by the user at the time of writing a program

ADVANTAGES OF FUNCTIONS

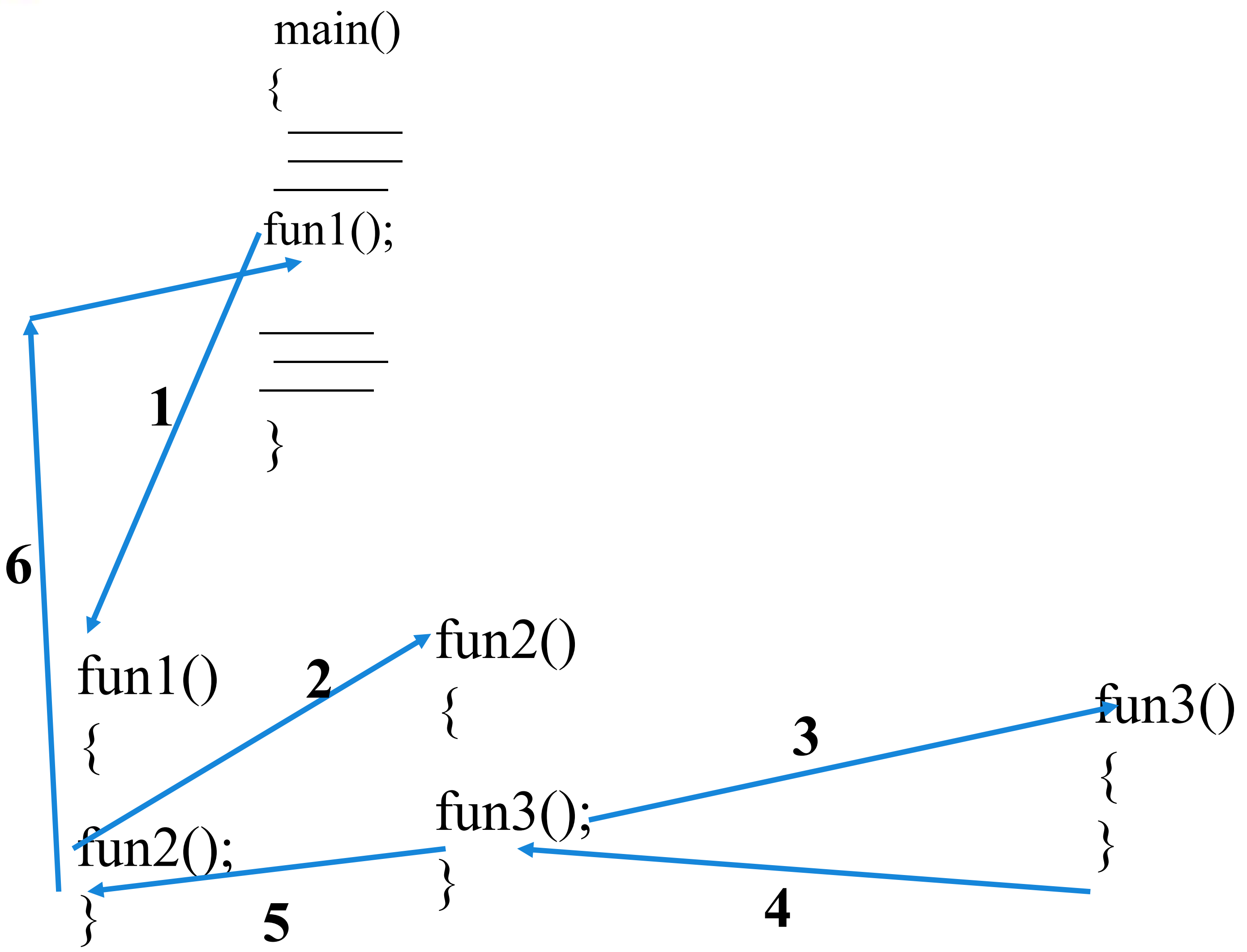
✓facilitates top-down modular programming

✓the length of a source program can be reduced by using functions at appropriate places

✓easy to locate and isolate a faulty function for further investigations

✓Avoids repetition of code

FLOW CONTROL IN A MULTIFUNCTION PROGRAM



```
main()
{
  ______
  ______
 ______
fun1();

______
 ______
______
}
```

```
fun1()
{
fun2();
}
```

```
fun2()
{
fun3();
}
```

```
fun3()
{
}
```

Functions in C

- A called function receives control from calling function.
- When the called function completes its task, it returns to calling function.
- It may or may not return a value to the caller.

User-Defined Functions

- **Function Declaration** means Specifying about our function to compiler.
- **Function Definition** means providing a body or behavior to the function.
- **Function calling** means invoking a function or sending a controller from calling function to called function.

- **FUNCTION PROTOTYPE DECLARATION**

Syntax:

```
return_type    function_name (arguments);
```

- **FUNCTION CALLING**

Syntax:

```
function_name (arguments);
```

- **FUNCTION DEFINITION**

Syntax:

```
return_type    function_name (arguments)
{
}
```

Functions can be classified into **4 types** based on arguments (Parameters) and return values

	Arguments	**Return Value**
Functions	Without	Without
Functions	Without	With
Functions	With	Without
Functions	With	With

Sample Function

```
int find_sum ( int num1 , int num2 ); //Prototype Declaration
int main()
{
            int result;
            printf(“Functions Demo \n”);
                        result = find_sum(10,20);  //calling
            printf(“ SUM = %d ”,result);
            return 0;
}
int find_sum ( int num1 , int num2 )
{
            int sum = num1 + num2;                //Definition
            return sum;
}
```

Formal and Actual Parameters

- Formal parameters are variables that are declared in the header of the function definition.
- Actual parameters are the expressions in the calling statement.
- The formal and actual parameter must match exactly in type, order and number. Their names, do not need to match.

Recursive function:

Example

```
int factorial(int number)
{
    if(number <= 0)
       return 1;
    return number * factorial(number - 1);
}
int main()
{
    int x = 3;
    printf("factorial of %d is %d",x,factorial(x));
 }
```