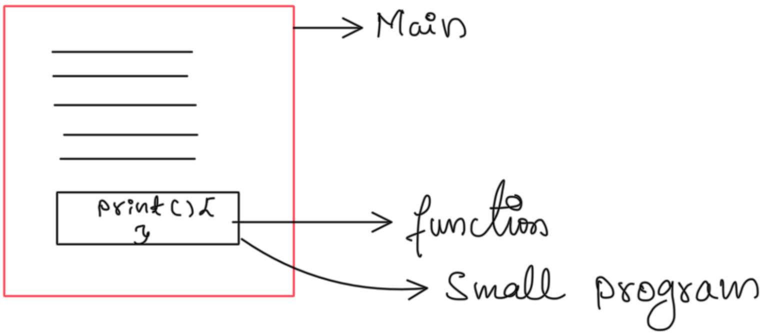
Functions

Day 12
22/07/2025

What is the function?
Handles Repetative task of Software Program



Makes code "Small precise and easily understandable"

Why functions?

① code is Reused, Time saving, Developer's money will be Saved. Saving Resources

② Readability, names given to functions increase code Readability

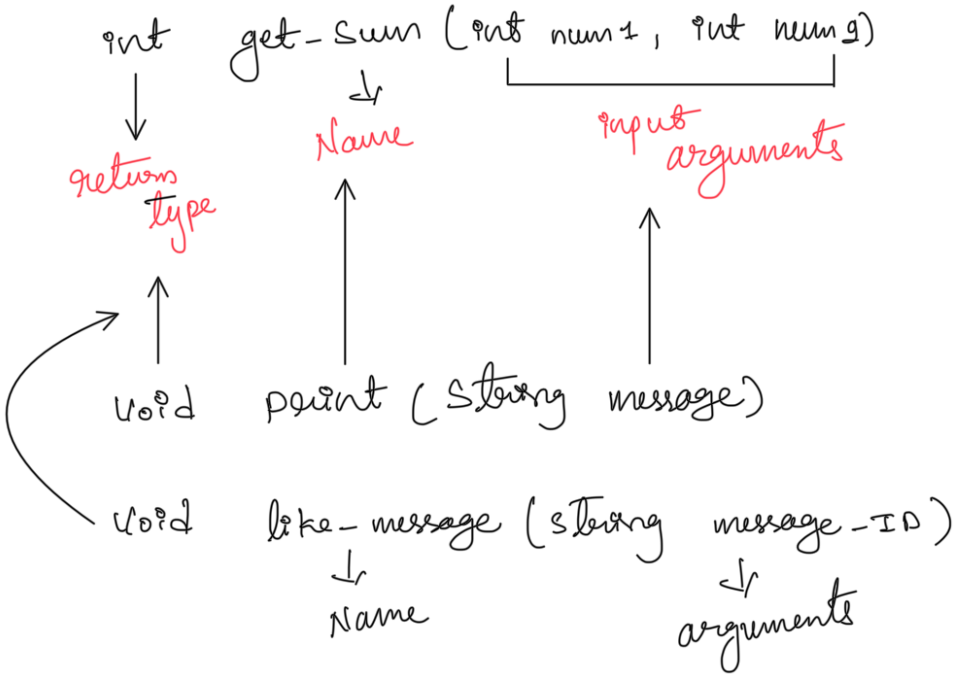
③ Function syntax

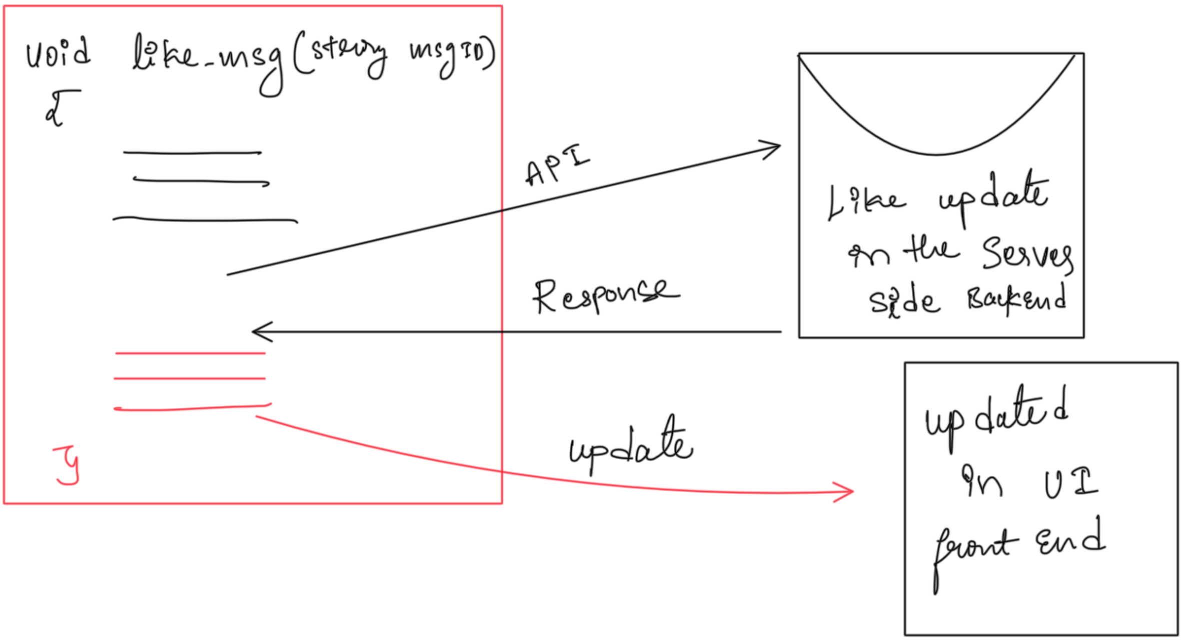
< Return type > function name (input arguments)
① ② ③
Mandatory

① Return Type → Return Type in function is not compulsory function can return Void (Nothing) after Execution.

② Input Arguments are also not compulsory. Some Cases function can Execute without taking arguments also arguments also

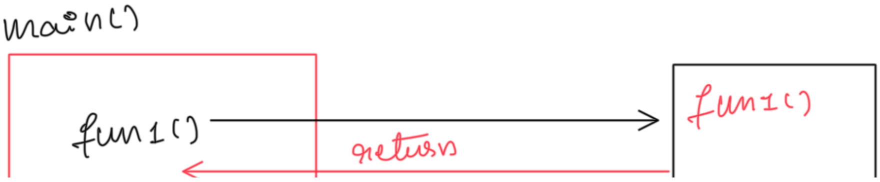
Examples

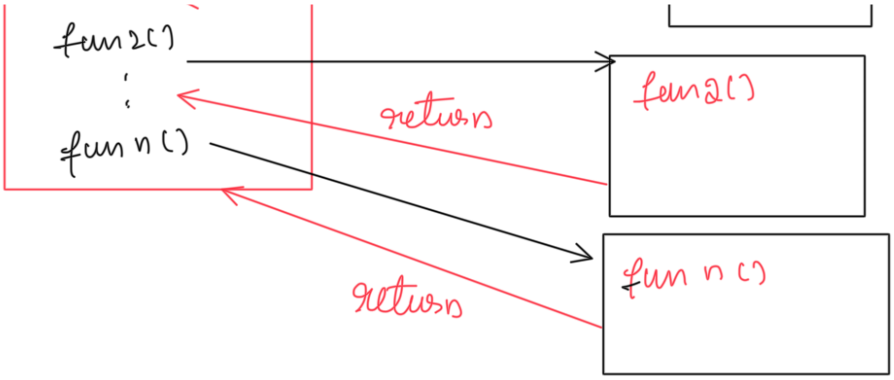

int get_sum (int num1, int num2)
return type
Name
input arguments
void print (string message)
void like_message (string message_ID)
Name
arguments


void like_msg(string msgID)
{
}
API
Response
update
Like update on the Server side Backend
updated in UI front End

In a software Internally Everything is a function

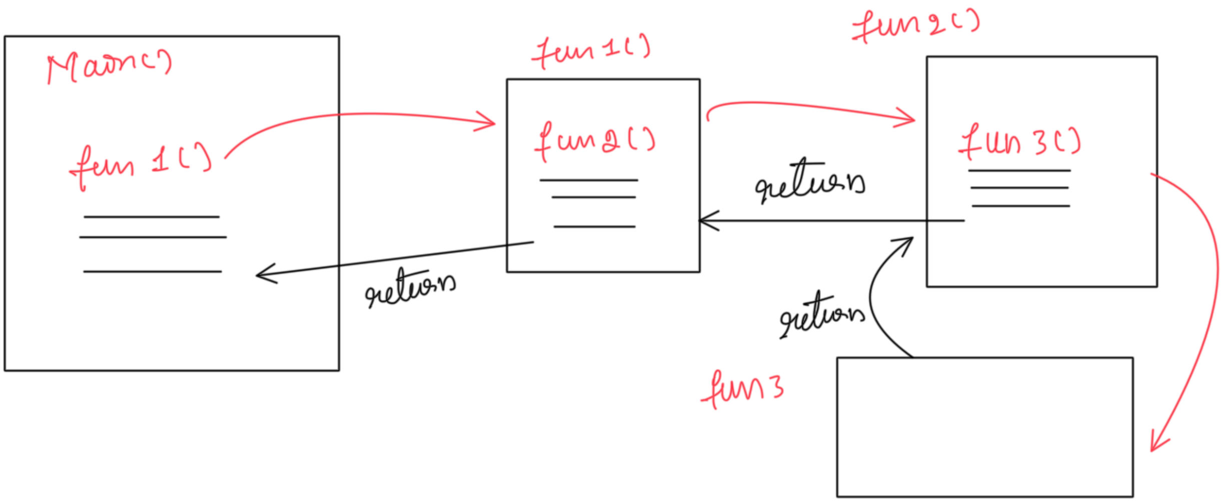
A function can call other functions It can call n number of other functions to perform task.



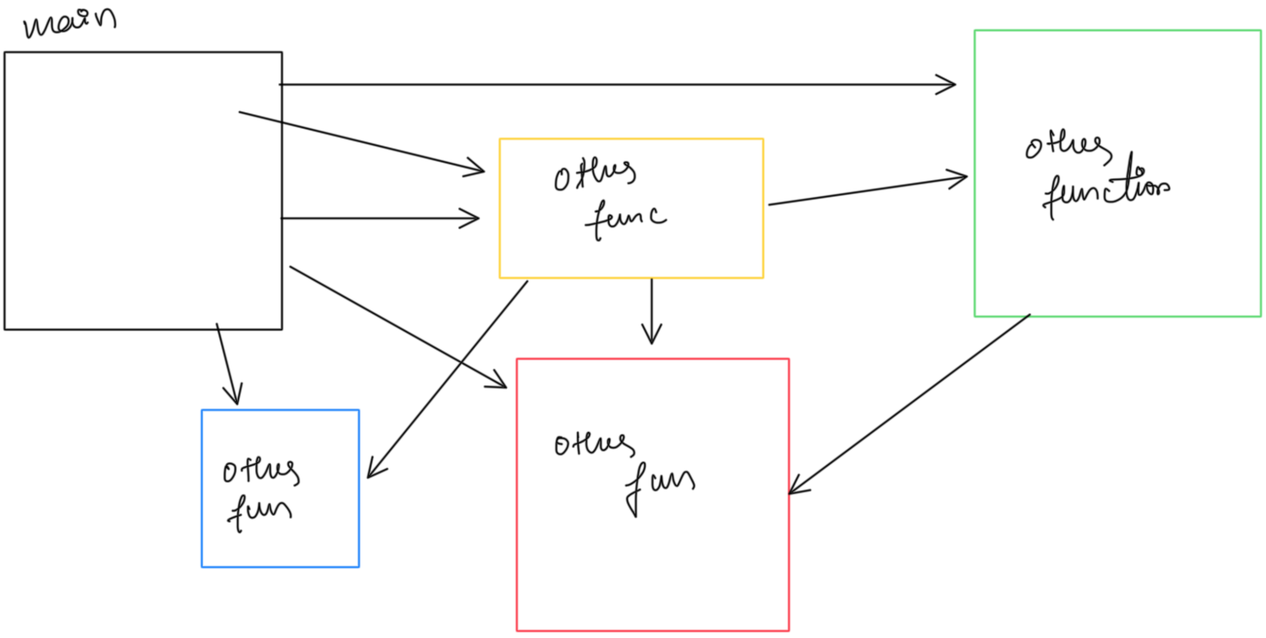


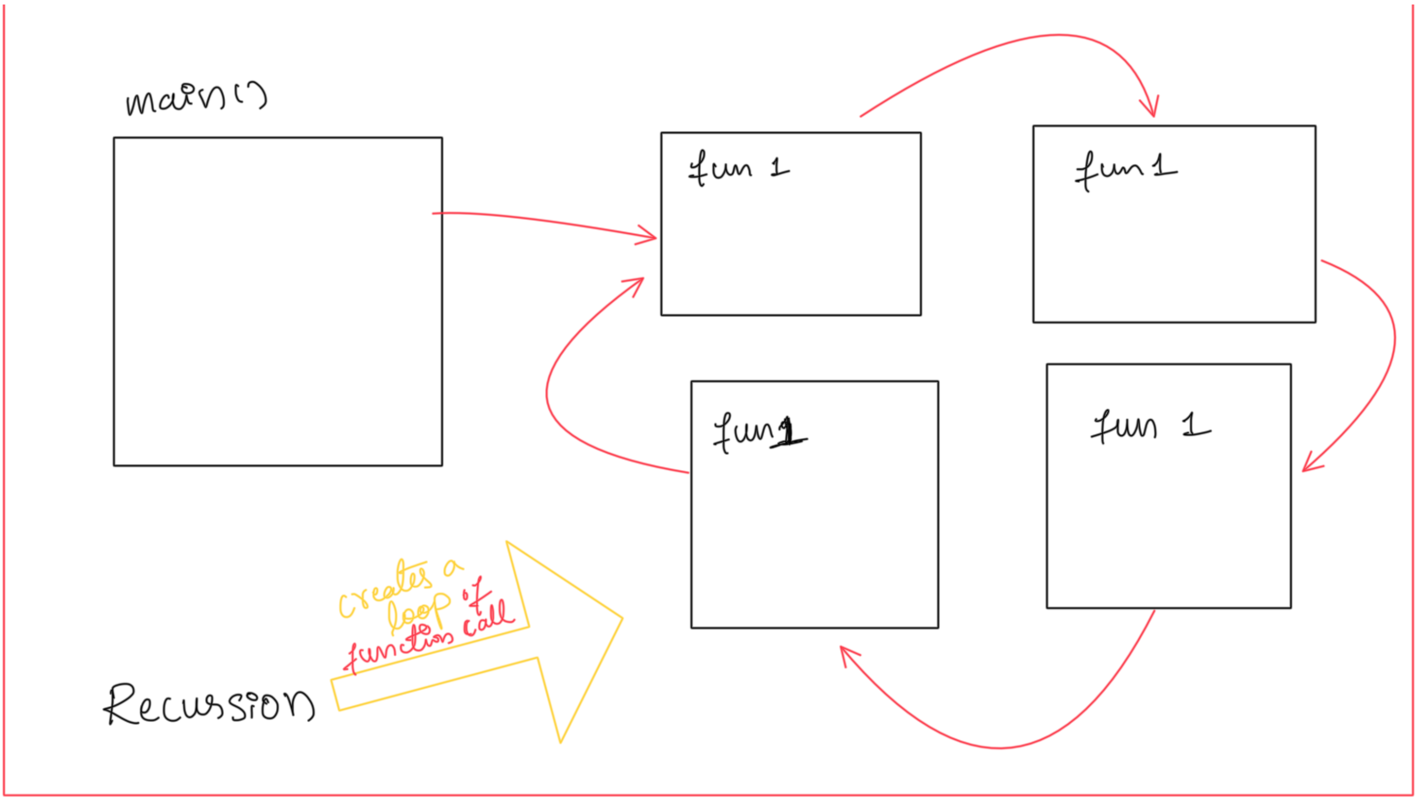
Another Type

Any called function can also call any other function



Any function can call any other functions for any number of times. (untill stack memory overflow)

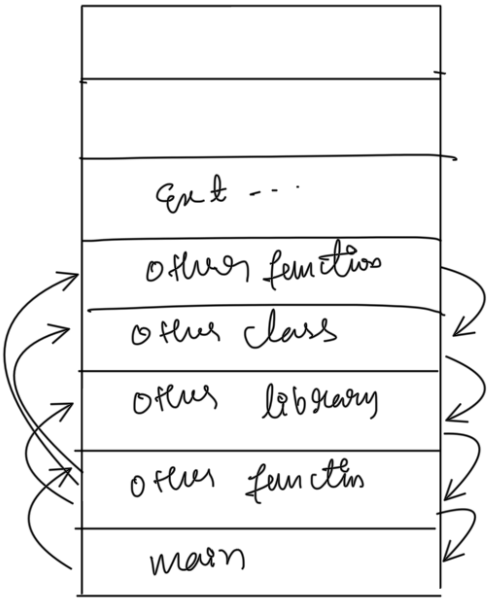




The numbers of function's can be called till stack memory over flow. (There is also a limit for function call's)

function call Heirarchy

function call stack



Ext ...

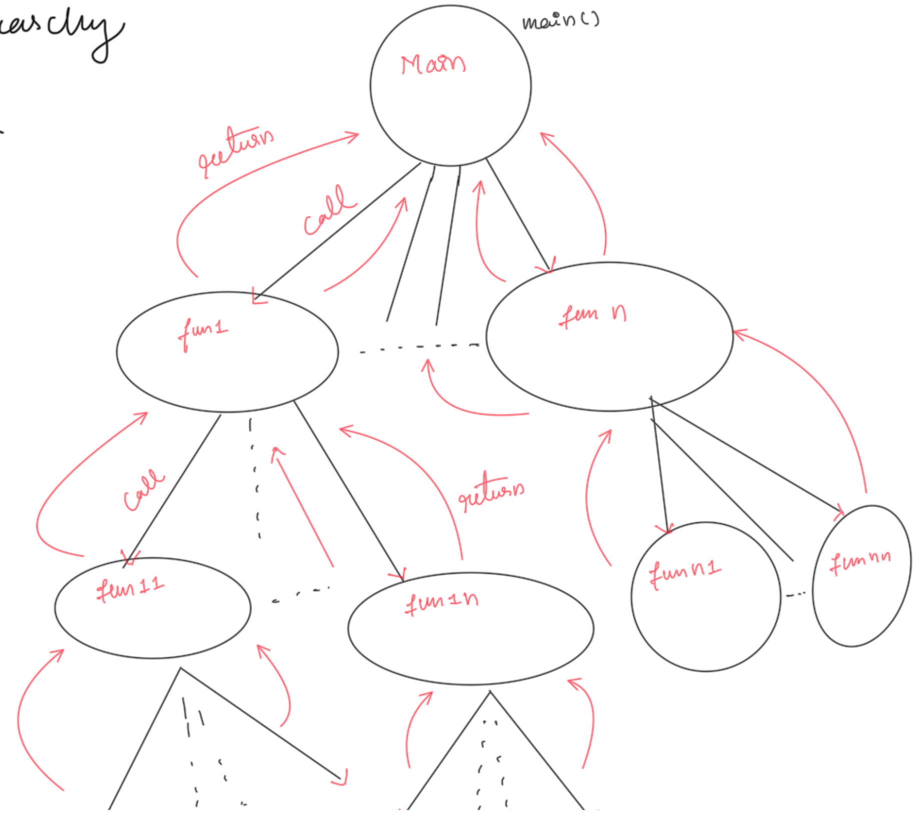
other function

other class

other library

other function

main



① function Declaration

Telling the compiler about function name & its return type & its arguments

② Defination

Writing code inside the function deciding the task of function inside

code writers task

③ Invocation / calling

calling the function to perform its tasks and Results calculation

function call can be performed By user & programmer

```
int    get_sum (int number1, int number2);
```

→ function declaration

```
int   get_sum  (int number1, int number2){

          int   sum = number1 + number2;
          return   sum;

   }
```

→ function defination

decides task

calculate result

calling

return

```
void main(){

   int ans = get_sum (10, 15);

}
```

function calling or Invocation

result returned

functions are Super Important for Interviews

What is main function ? How it is used

```
int ~~main~~ () {        →  program will not Run
                            without "Main" function
    printf ("Hello");

}
```

The first function which is invoked when the program or Software is start Executing. → main function

starting point of software Execution

Main is also a function we can also pass Parameters to main function.

Main function → takes input in strings format

main function can take n Numbes of arguments in string format.

Command Line argument

passing argument to main function through Command line

Hello.Exe Veeresh → command Line Argument

Hello.Exe → file name table

Hello.Exe → main (Veeresh) → printf "Hello Veeresh"

Executa
file.

command line Arguments are Super Important

Main function is required to Run the program

Flowchart: Command Line Arguments in C

In C, command line arguments are passed through the main function:

int main(int argc, char *argv[])

Where:

- argc: Number of arguments passed

- argv: Array of arguments (strings)

Flowchart Steps:

1. Start

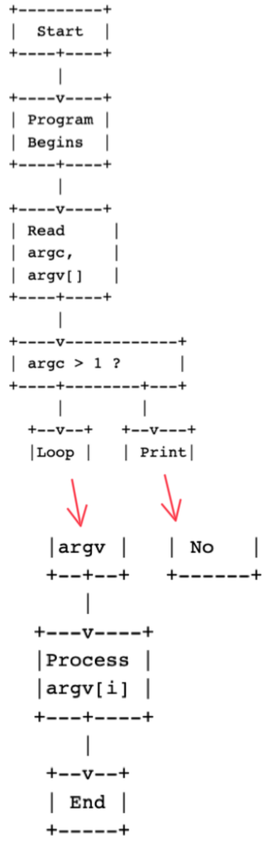
2. Program Execution Begins

3. Read argc and argv[]

4. Check if argc > 1

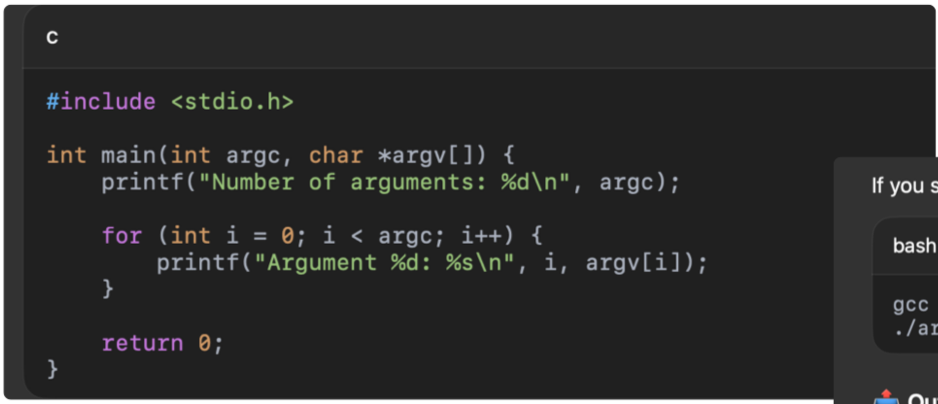
   - If Yes: Loop through argv[1] to argv[argc-1] and process

   - If No: Print 'No arguments passed'



```
+---------+
|  Start  |
+----+----+
     |
+----v----+
| Program |
| Begins  |
+----+----+
     |
+----v----+
| Read     |
| argc,    |
| argv[]   |
+----+----+
     |
+----v------------+
| argc > 1 ?       |
+----+--------+---+
     |        |
  +--v--+   +--v---+
  |Loop |   | Print|

  |argv |   | No    |
  +--+--+   +------+
     |
 +---v----+
 |Process |
 |argv[i] |
 +---+----+
     |
  +--v--+
  | End |
  +-----+
```



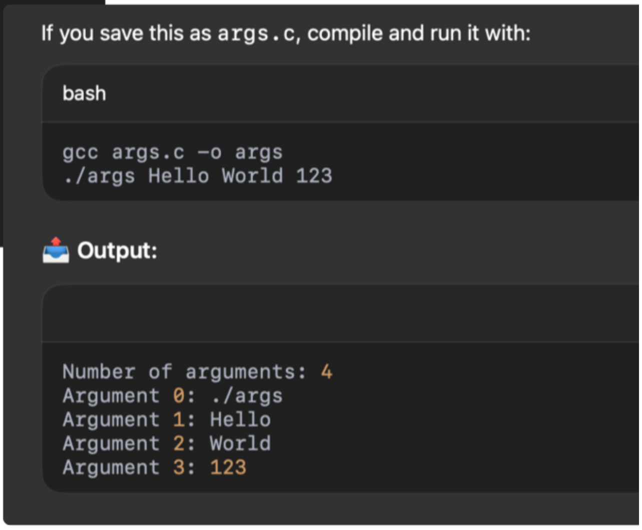
c

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Number of arguments: %d\n", argc);

    for (int i = 0; i < argc; i++) {
        printf("Argument %d: %s\n", i, argv[i]);
    }

    return 0;
}
```



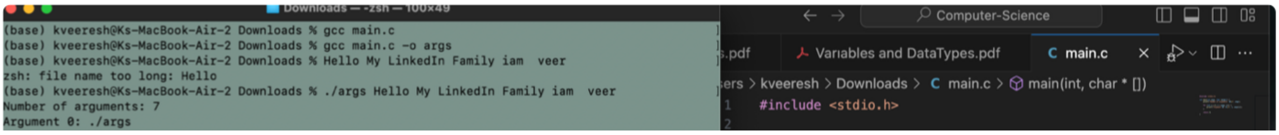
If you save this as `args.c`, compile and run it with:

bash

```bash
gcc args.c -o args
./args Hello World 123
```

📤 **Output:**

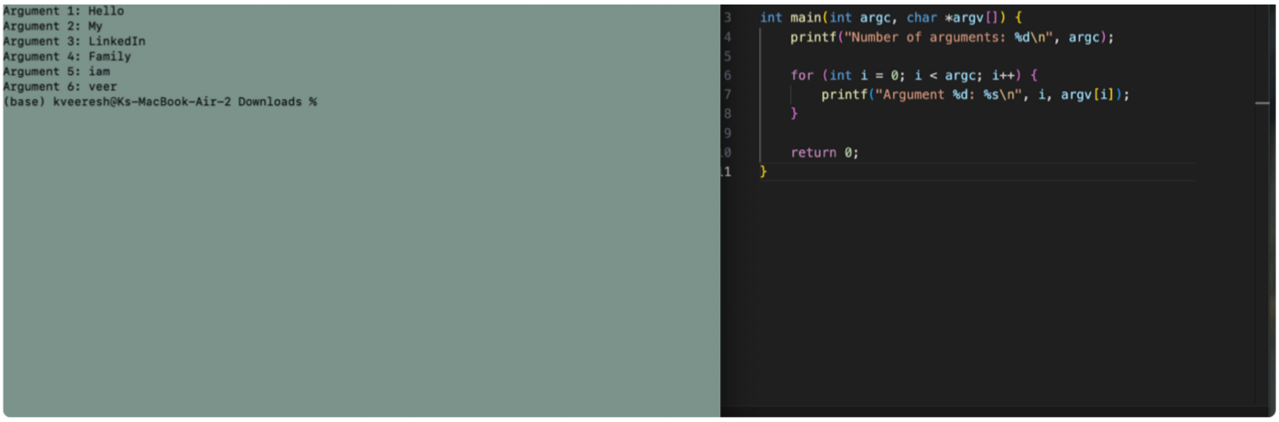
```
Number of arguments: 4
Argument 0: ./args
Argument 1: Hello
Argument 2: World
Argument 3: 123
```



Downloads — -zsh — 100×49
Computer-Science
.pdf
Variables and DataTypes.pdf
main.c
ers > kveeresh > Downloads > main.c > main(int, char * [])

```
(base) kveeresh@Ks-MacBook-Air-2 Downloads % gcc main.c
(base) kveeresh@Ks-MacBook-Air-2 Downloads % gcc main.c -o args
(base) kveeresh@Ks-MacBook-Air-2 Downloads % Hello My LinkedIn Family iam  veer
zsh: file name too long: Hello
(base) kveeresh@Ks-MacBook-Air-2 Downloads % ./args Hello My LinkedIn Family iam  veer
Number of arguments: 7
Argument 0: ./args
```

```c
#include <stdio.h>
```

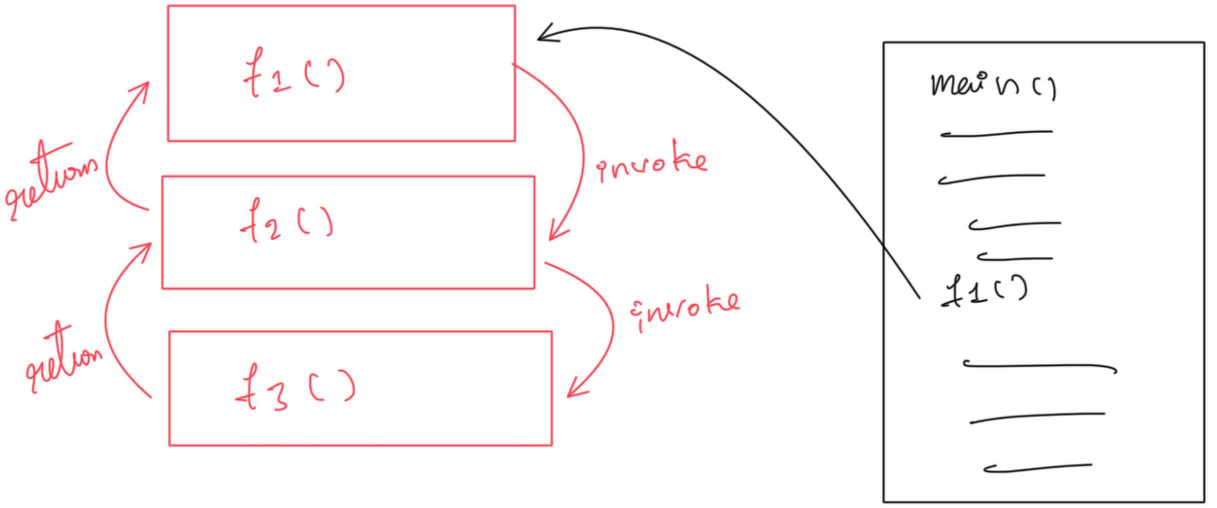


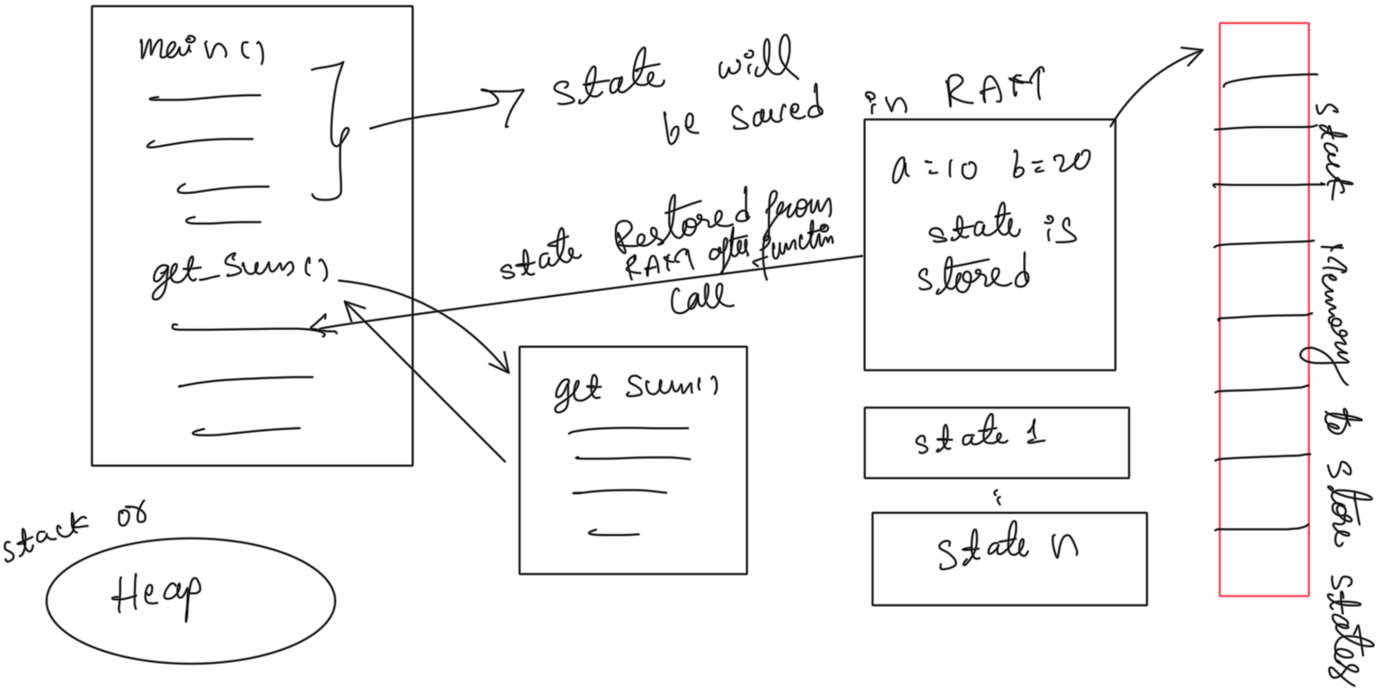
```
Argument 1: Hello
Argument 2: My
Argument 3: LinkedIn
Argument 4: Family
Argument 5: iam
Argument 6: veer
(base) kveeresh@Ks-MacBook-Air-2 Downloads %
```

```c
int main(int argc, char *argv[]) {
    printf("Number of arguments: %d\n", argc);

    for (int i = 0; i < argc; i++) {
        printf("Argument %d: %s\n", i, argv[i]);
    }

    return 0;
}
```




main()
get_Sum()
state will be saved in RAM
state Restored from RAM after function call
a=10 b=20
state is stored
get Sum()
state 1
:
state n
stack Memory to store states
stack or
Heap

Local variables of a function are stored in Stack memory of the RAM after function execution Memory data is cleared.

Understanding Functions in Programming

What is a Function?

A function is a reusable block of code that performs a specific task. It helps divide a program into smaller and modular chunks.

Why Functions?

- Code Reusability
- Improves readability and modularity
- Easier to debug and maintain
- Avoids repetition

Function Syntax (C-like Languages):

Function Declaration:

```
    return_type function_name(parameter_list);
```

Function Definition:

```
    return_type function_name(parameter_list) {
        // body
    }
```

Function Invocation:

```
    function_name(arguments);
```

Example in C:

```
#include <stdio.h>

int add(int a, int b);  // Declaration

int main() {
    int result = add(5, 3);  // Invocation
    printf("%d", result);
    return 0;
}

int add(int a, int b) {  // Definition
    return a + b;
}
```

Function vs Method (OOP Context):

- Function: Defined outside class (C/C++)
- Method: Defined inside class (Java, Python)

Main Function:

Every C/C++ program must have a main() function as the entry point.

Command Line Arguments:

```
int main(int argc, char *argv[]) {
    // argc: number of arguments
    // argv: array of arguments
}
```

Why Python Doesn't Have main() by Default?

Python is a scripting language. The code runs top-down. However, a main() function can be defined for clarity.

Python Example:

```
def main():
    print("Hello")

if __name__ == "__main__":
    main()
```

Flowchart:

[Start] --> [Function Declaration] --> [Function Definition] --> [Invoke Function] --> [Output] --> [End]

Interview Questions:

----

1. What are functions and why are they used?

2. Difference between function declaration and definition?

3. What is the role of the main function?

4. How do command line arguments work in C?

5. Why does Python allow top-down execution?