A job ready bootcamp in C++, DSA and IOT

Function

Saurabh Shukla (MySirG)

Agenda

1. What is a function
2. Function call and Function definition
3. Predefined and User defined function
4. Flow of program with multiple functions
5. Benefits of functions
6. Ways to define a function
7. Header files and Library files

What is a function?

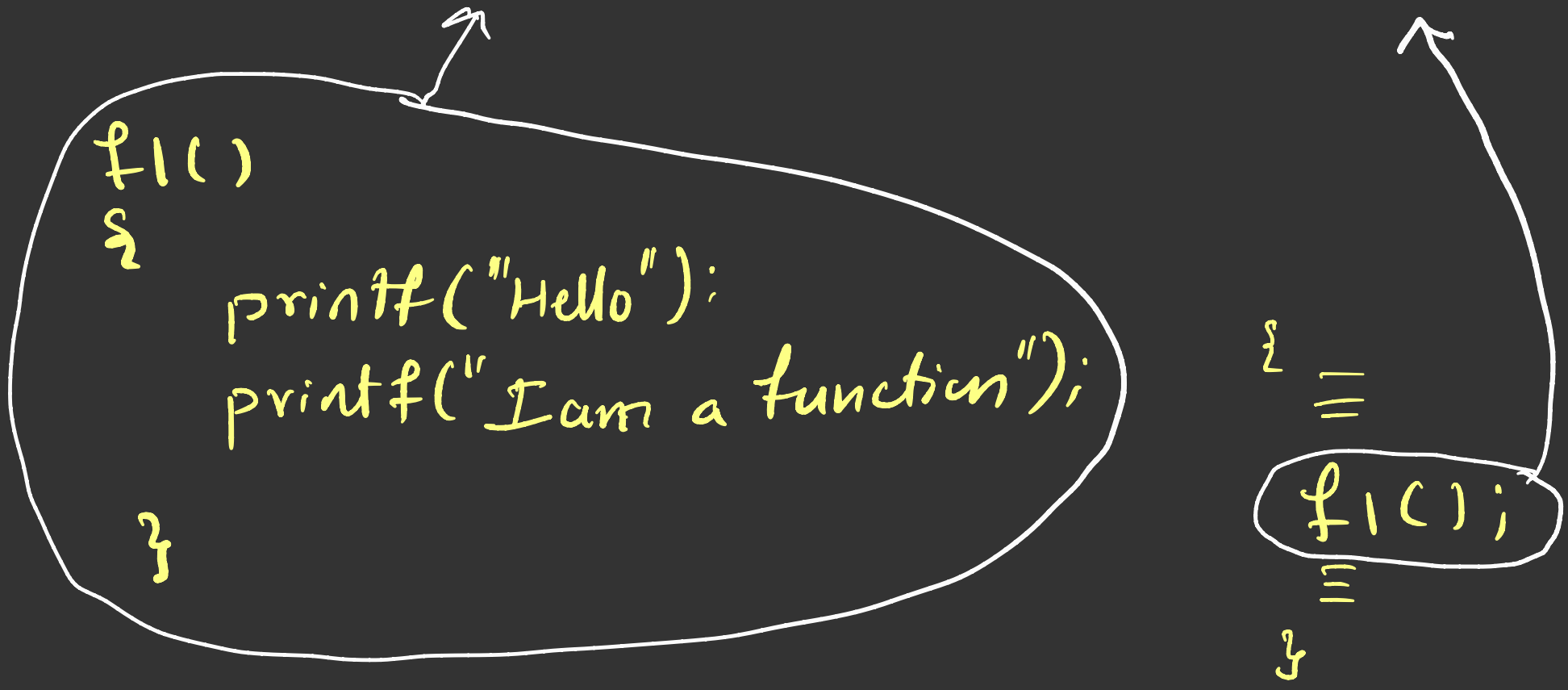
- Function is a basic building block of a C program.
- Function is a block of code, which has some name for identification.
- A C program can have any number of functions

```
functionName( )
{
        // some code

}
```

Function Definition

- Function names must be unique in a program
- Function definition vs function call



- Function is a way to implement modularization.
- Modularization is splitting up of a bigger task into several smaller sub-tasks to reduce the complexity of a problem.
- You can compile a C file without having main() function but cannot run.

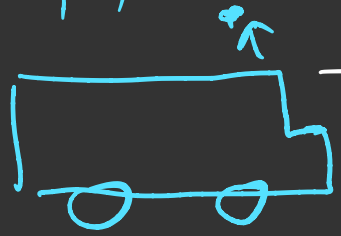
Functions are of two types

1. Predefined Functions printf() scanf()
2. User defined Functions

```
main()
{
    ---
    ---
    ---
    ---
}
```

switch() ✗
while() ✗
if() ✗
sizeof() ✗
return(0) ✗
clrscr() ✓

Flow of a program



```
main()
{
    a();
    b();
    a();
}
a()
{
    printf("Hello");
}
b()
{
    printf("Bye");
    a();
}
```

Program's memory

printf()

Hello Bye Hello Hello

Benefits of functions

1. Easy to Read
2. Reduce complexity
3. Easy to modify
4. Easy to debug
5. Code reusability
6. Avoids rewriting
7. Better memory utilization

```
a()
{
    int x;

}

b()
{
    int y;

}
```

Ways to define a function

1. Takes Nothing, Returns Nothing
2. Take something, Returns Nothing
3. Takes Nothing, Return Something
4. Take something, Return Something

TNRN

```
#include <stdio.h>

void add();                 <-- Function Declaration or Function Prototype
int main()
{
    add();                  <-- Function call
    return 0;
}
void add()                  <-- TN
{
    int a,b,c;
    printf("Enter two numbers");
    scanf("%d%d",&a,&b);
    c=a+b;
    printf("Sum is %d",c);
}
```

Function Definition

no return keyword = RN

TSRN

```
void add(int, int);
int main()
{
    int x, y;
    printf("Enter two numbers");
    scanf("%d %d", &x, &y);

    add(x, y);
    return 0;
}
void add(int a, int b)
{
    int c;

    c = a+b;
    printf("Sum is %d", c);


}
```

← Actual Argument

← Formal Arguments

TNRS

```
int add( );
int main()
{
    int s;
    s=add();
    printf("Sum is %d", s);
    return(0);
}
int add()
{
    int a, b, c ;
    printf("Enter two numbers");
    scanf("%d%d",&a,&b);
    c=a+b;
    return c;
}
```

TSRS

```
int add(int,int);
int main()
{
    int s, x, y;
    printf("Enter two numbers");
    scanf("%d %d", &x, &y);
    s = add(x, y);
    printf("Sum is %d", s);
    return 0;
}
int add( int a, int b)
{
    int c;
    c = a+b;
    return c;
}
```

← Function call by passing values. (call by value)

Ask Yourself

1. Function call vs Function Definition vs Function Declaration
2. Function saves memory
3. main() is a userdefined function.
4. Header file vs Library File
5. When to write void?
6. When to write return?
7. Return type of main()
8. call by value
9. Function Prototype
10. Can we call main() function()?