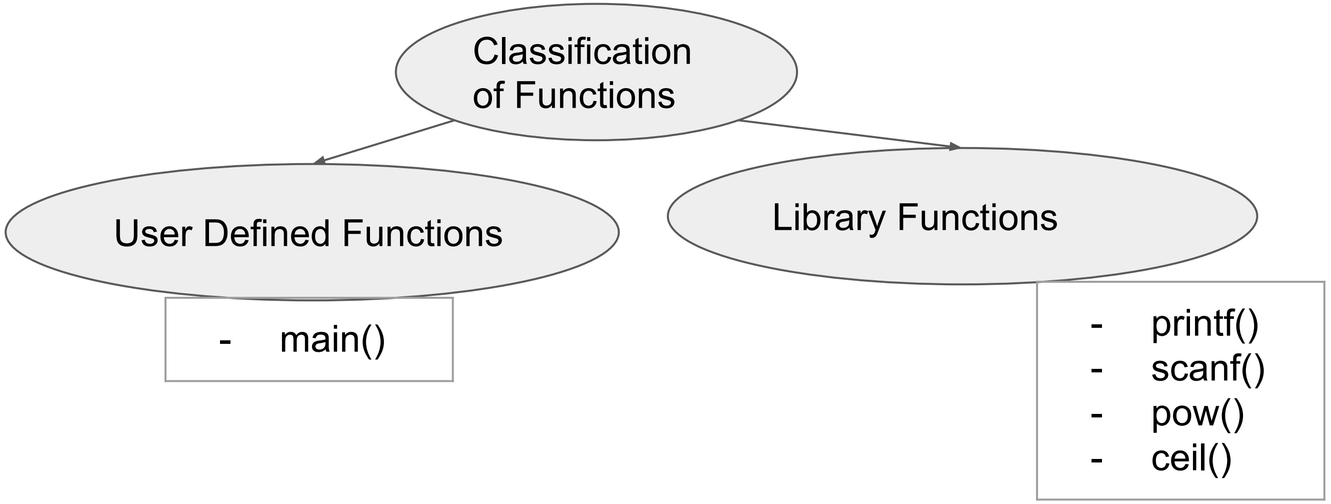
Functions in C

Prof. Harish D.G.
Dept. of Computer and IT
College of Engineering,Pune
www.harishgadade.com

What is function?

- A large program can be divided into many subprogram, that subprogram is called functions.
- Basically a job of function is to do something.
- Subprogram is a self-contained block and have a well defined purpose.
- C program contains at least one function which is main().



Advantages of a Function

- It is much easier to write a structured program where a large program can be divided into smaller, simpler task.
- Allowing code to be called many times.
- Easier to read and update.
- It is easy to debug and fix up the errors

Steps to write a function

- Declaration of a function
- Calling of a function
- Definition of a function

```
#include<stdio.h>
void hello();

void main()
{
        printf("First function program\n");
        hello();
}

void hello()
{
        printf("Hello World!!");
}
```

Function Declaration

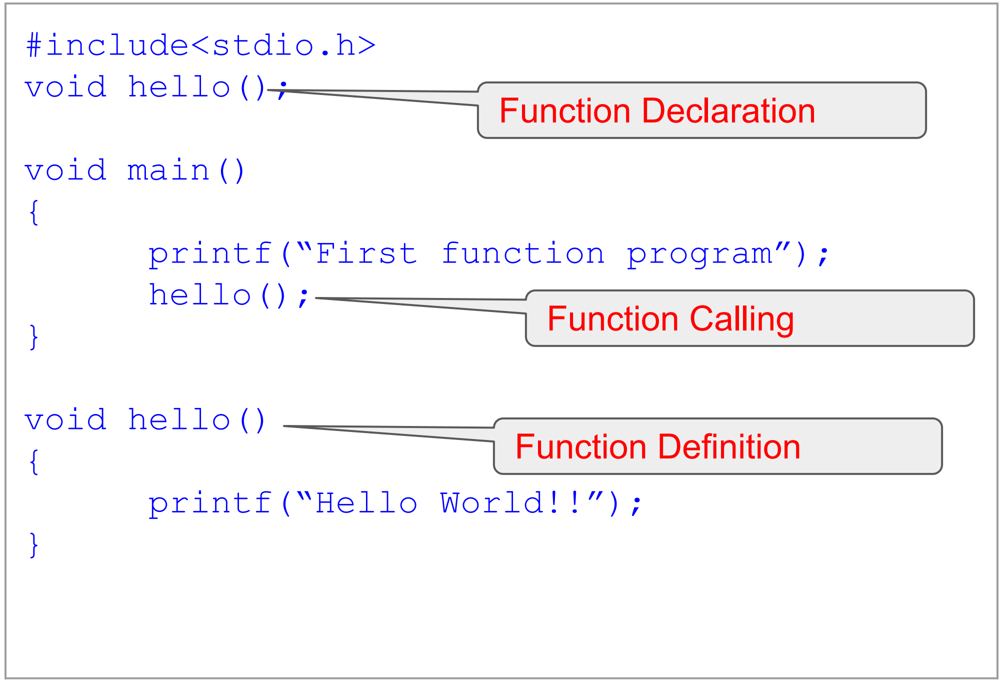
Function Calling

Function Definition

Types of Function

- Function with no arguments and no return value
- Function with no arguments and a return value
- Function with argument or arguments and no return value
- Function with arguments and a return value

Function with no arguments and no return value



```
#include<stdio.h>
void hello();

void main()
{
        printf(“First function program”);
        hello();
}

void hello()
{
        printf(“Hello World!!”);
}
```

Function with no arguments and a return value

```
#include<stdio.h>
int add();
```

Function Declaration

```
void main()
{
        int c;
        c=add();
```

Function Calling

```
        printf("Addition = %d\n",c);
}

int add()
```

Function Definition

```
{
        int a=10,b=20,c;
        c=a+b;
        return(c);
}
```

Output:

```
$ gcc sample.c
$ ./a.out
Addition = 30
```

Function with arguments and no return value

```
#include<stdio.h>
void add(int a,int b);

void main()
{
      int a=10,b=20;
      add(a,b);
}

void add(int a,int b)
{
      int c;
      c=a+b;
      printf("Addition = %d\n",c);
}
```

Output:

```
$ gcc sample.c
$ ./a.out
Addition = 30
```

Function with arguments and a return value

```c
#include<stdio.h>
int add(int a,int b);

void main()
{
      int a=10,b=20,c;
      c=add(a,b);
      printf("Addition = %d\n",c);
}

int add(int a,int b)
{
      int c;
      c=a+b;
      return(c);

}
```

Output:

```
$ gcc sample.c
$ ./a.out
Addition = 30
```

Local,Global and Static Variables

- **Local Variables**
    - Variables that are declared inside a function or block are called local variables
    - They can be used only by statements that are inside that function or block of code.
    - Local variables are not known to functions outside their own.

- **Global Variables**
    - Global variables are defined outside a function, usually on top of the program.
    - Global variables hold their values throughout the lifetime of your program and they can be accessed inside any of the functions defined for the program.
    - A global variable can be accessed by any function.

Local,Global and Static Variables

```c
#include <stdio.h>
int main ()
{
  /* local variable declaration */
  int a, b;
  int c;

  /* actual initialization */
  a = 10;
  b = 20;
  c = a + b;

  printf ("value of a = %d\n”,a);
  printf ("value of b = %d\n”,b);
  printf ("value of c = %d\n”,c);

  return 0;
}
```

```
Output:

$ gcc sample.c
$ ./a.out

value of a = 10
value of b = 20
value of c = 30
```

Local,Global and Static Variables

```
#include <stdio.h>
/* global variable declaration */
int g;
int main ()
{
  /* local variable declaration */
  int a, b;

  /* actual initialization */
  a = 10;
  b = 20;
  g = a + b;

  printf ("value of a = %d\n”,a);
  printf ("value of b = %d\n”,b);
  printf ("value of g = %d\n”,g);
  return 0;
}
```

```
#include <stdio.h>

/* global variable declaration */
int g = 20;

int main ()
{
  /* local variable declaration */
  int g = 10;

  printf ("value of g = %d\n",  g);

  return 0;
}
```

Static Variables

- Static variables have a property of preserving their value even after they are out of their scope and are not initialized again in the new scope.
- Static variable is declared as a static by writing the keyword static in front of variable declaration.

```
void main()
{
     int a;
}
```

```
void main()
{
     Static int a;
}
```

- Default Value of Static variable is 0
- Scope of variable is life time in program.

Static Variables

```c
#include<stdio.h>
int fun()
{
  int count = 0;
  count++;
  return count;
}

int main()
{
  printf("%d ", fun());
  printf("%d ", fun());
  return 0;
}
```

```c
#include<stdio.h>
int fun()
{
  static int count = 0;
  count++;
  return count;
}

int main()
{
  printf("%d ", fun());
  printf("%d ", fun());
  return 0;
}
```

Static Variables

```
#include<stdio.h>
int fun()
{
  int count = 0;
  count++;
  return count;
}

int main()
{
  printf("%d ", fun());
  printf("%d ", fun());
  return 0;
}

Output:
1 1
```

```
#include<stdio.h>
int fun()
{
  static int count = 0;
  count++;
  return count;
}

int main()
{
  printf("%d ", fun());
  printf("%d ", fun());
  return 0;
}
```

Static Variables

```c
#include<stdio.h>
int fun()
{
  int count = 0;
  count++;
  return count;
}

int main()
{
  printf("%d ", fun());
  printf("%d ", fun());
  return 0;
}
```

```
Output:
1 1
```

```c
#include<stdio.h>
int fun()
{
  static int count = 0;
  count++;
  return count;
}

int main()
{
  printf("%d ", fun());
  printf("%d ", fun());
  return 0;
}
```

```
Output:
1 2
```

Recursive Function

- A function that calls itself is known as a recursive function

```
void main()
{
        …….
        …….
        rec_fun();
        …….
        …….
}

void rec_fun()
{
        ……
        ……
        rec_fun();
        ……
        ……
}
```

```
void main()
{
        printf("\n Recursion Program");
        main();
}
```

Output:

```
Recursion Program
Recursion Program
Recursion Program
Recursion Program
:
:


Execution will continue indefinitely
```

Recursive Function

- A function that calls itself is known as a recursive function

```
#include <stdio.h>
int factorial(int);
int main()
{
   int n=5,fact;
   fact=factorial(n);
   printf("\n Factorial=%d",fact);
   return 0;
}
int factorial(int n)
{     if(n==1)
          return(1);
     else
          return(n*factorial(n-1));
}
```

Output :

```
Factorial= 120
```

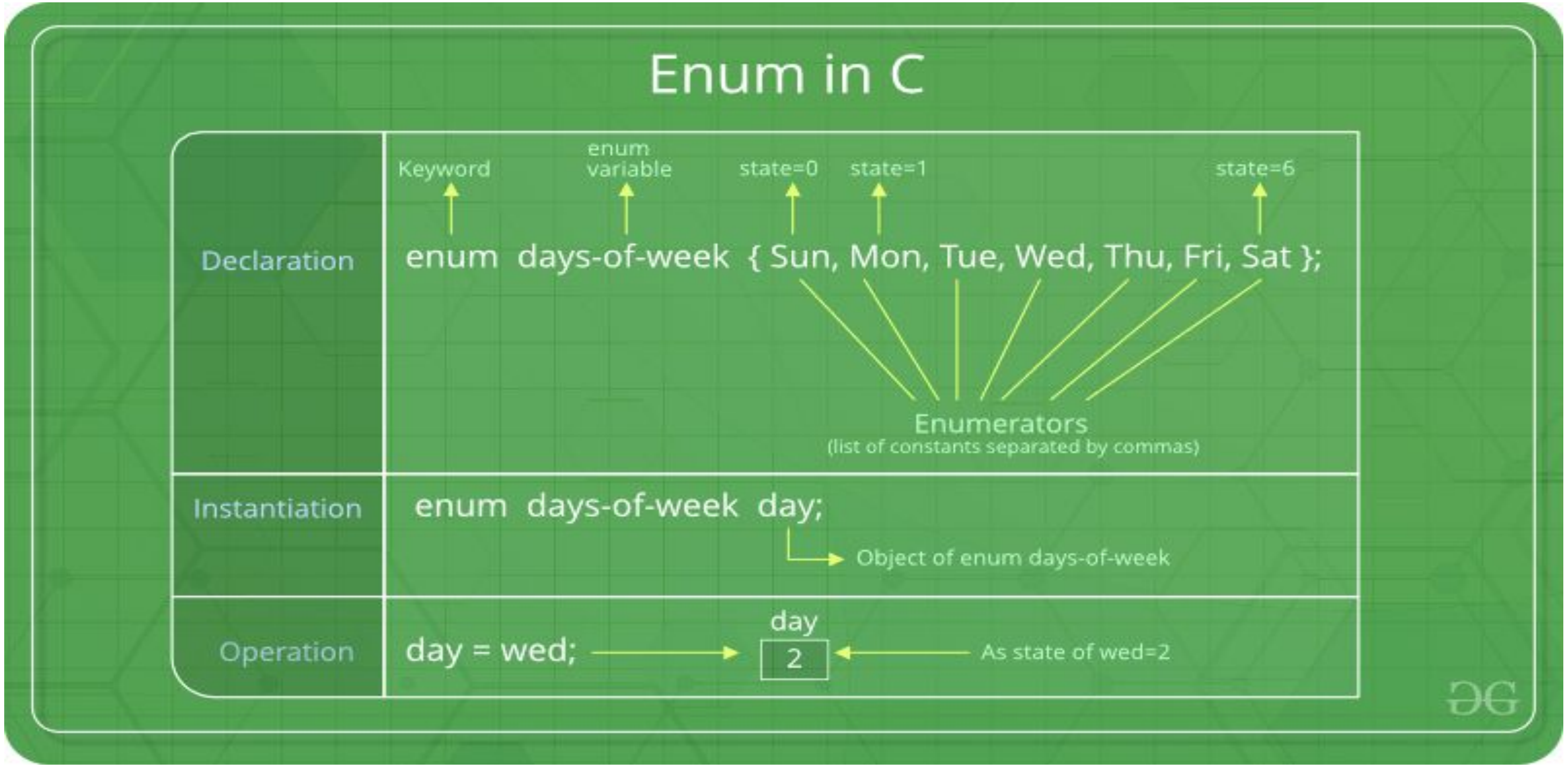
User Defined Data Types

- The data types that are defined by the user are called the derived data types
- The User Defined Data Types are;
    - Class in C++
    - Structure
    - Union
    - typedef
    - Enum

User Defined Data Types

- The data types that are defined by the user are called the derived data types
- The User Defined Data Types are;
  - Class in C++
  - Structure
  - Union
  - typedef
  - Enum : It is mainly used to assign names to integral constants.

Enum Data Types



Enum Data Types

```
#include<stdio.h>

enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};

int main()
{
    enum week day;
    day = Wed;
    printf("%d",day);
    return 0;
}
```

Output:

```
2
```