enum

- enum is user defined data type.
- Used to improve readability of C program (int constants, switch-case constants).

```
enum color { RED, GREEN, BLUE, WHITE, YELLOW };
enum color c1 = BLUE;
```

- enum constant values by default start from 0 and assigned sequentially.
- Programmer may choose to modify enum constant to any +ve, 0 or –ve value.
- Enum constants can be duplicated.

```
enum color { RED=-2, GREEN, BLUE, WHITE, YELLOW=0 };
```

- Internally enum is integer, so size of enum = size of int.
- The enum constants are replaced by int values.

typedef

- typedef is used to create alias for any data-type.
- These aliases are helpful to
- increase readability of the code.
- port same code across multiple architecture/platforms.
- simplify complex declarations.
- syntax

```
typedef existing-data-type data-type-alias;
```

- Examples:

```
typedef char int8_t;
typedef unsigned char uint8_t;
typedef unsigned int size_t; // declared in C library.
```

Functions

- C program is made up of one or more functions.
- C program contains at least one function i.e. main() function.
- Execution of C program begins from main.

- It returns exit status to the system.
- Advantages
  - Reusability
  - Readability
  - Maintainability
- Function is set of instructions, that takes zero or more inputs (arguments) and return result (optional).
- Function is a black box.
- Each function has
  - Declaration

```
<return type> <function name> (<list of type of arguments>);
```

  - Definition

```
<return type> <function name> (<list of arguments>)
{
    //body
}
```

  - Call

```
<function name>(<list of arguments>)
```

- A function can be called one or more times.
- Arguments
  - Arguments passed to function Actual arguments
  - Arguments collected in function Formal arguments
  - Formal arguments must match with actual arguments

Function Declaration

- Informs compiler about function name, argument types and return type.
- Usually written at the beginning of program (source file).
- Can also be written at start of calling function).
- Examples:

```
float divide(int x, int y);
int fun2(int, int);
```

```
int fun3();
double fun4(void);
void fun5(double);
```

- Declaration statements are not executed at runtime.

Function Definition

- Implementation of function.
- Function is set of C statements.
- It process inputs (arguments) and produce output (return value).
- Example

```
float divide(int a, int b) {
    return (float)a/b;
}
```

- Function can return max one value.
- Function can be defined in another function.

Function Call

- Typically function is called from other function one or more times.

Function Execution

- When a function is called, function activation record/stack frame is created on stack of current process.
- When function is completed, function activation record is destroyed.
- Function activation record contains:
  - Local variables
  - Formal arguments
  - Return address
- Upon completion, next instruction after function call continue to execute.

Function Types

- User defined functions
  - Declared by programmer
  - Defined by programmer
  - Called by programmer
- Library (pre-defined) functions
  - Declared in standard header files e.g. stdio.h, string.h, math.h, ...
  - Defined in standard libraries e.g. libc.so, libm.so, ...
  - Called by programmer
- main()
  - Entry point function (code perspective)
  - User defined

    - System declared
    - int main(void) {...}
    - int main(int argc, char *argv[]) {...}

Storage classes

- Each running process have following sections:
    - Text
    - Data
    - Heap
    - Stack
- Storage class decides
    - Storage (section)
    - Life (existence)
    - Scope (visibility)
- Accessing variable outside the scope raise compiler error.
- Local variables declared inside the function.
    - Created when function is called and destroyed when function is completed.
- Global variables declared outside the function.
    - Available through out the execution of program.
    - Declared using extern keyword, if not declared within scope.
- Static variables are same as global with limited scope.
    - If declared within block, limited to block scope.
    - If declared outside function, limited to file scope.
- Register is similar to local storage class, but stored in CPU register for faster access.
    - register keyword is request to the system, which will be accepted if CPU register is available.