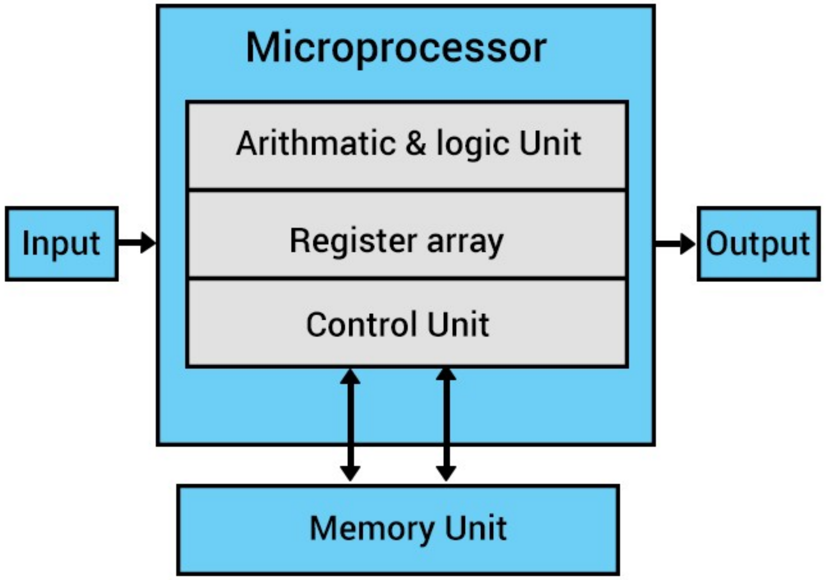
CSE 211: Introduction to Microprocessors

Tutorial 1

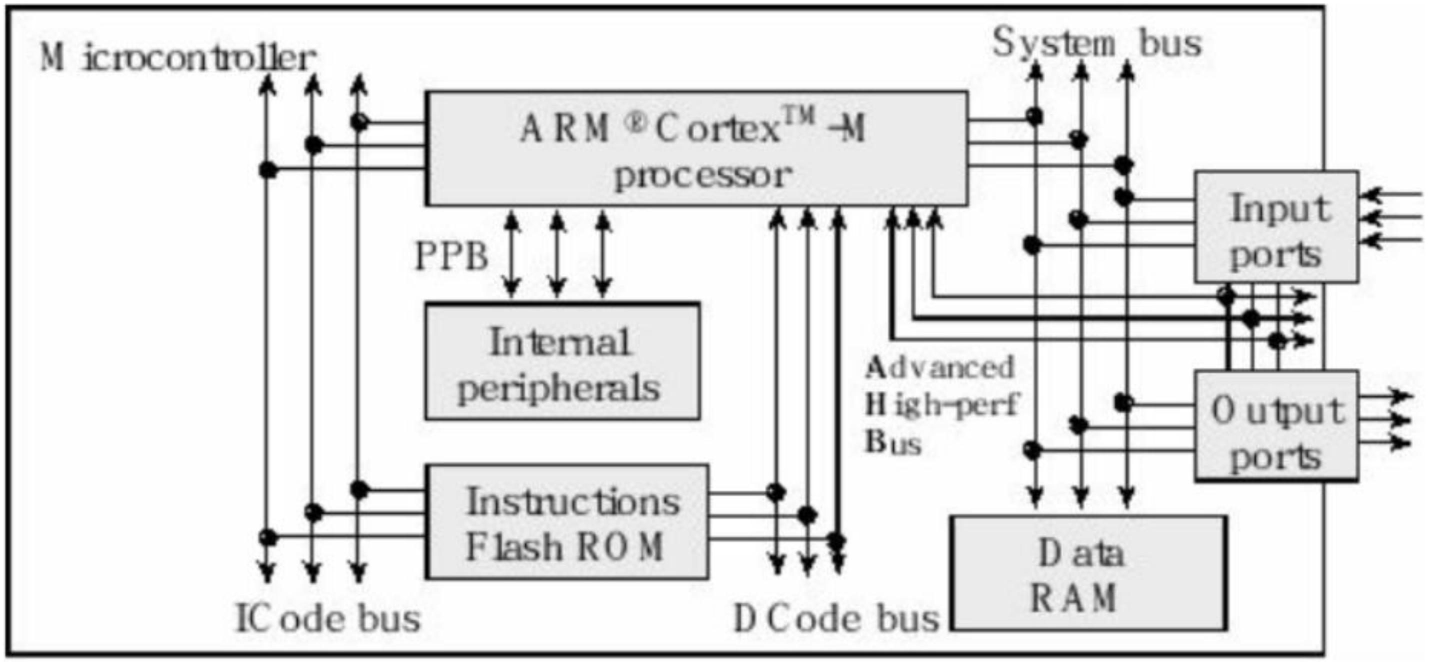
Coursework

Midterm	20 marks
Project	10 marks
2 Quizzes	5 marks
Attendance	5 marks

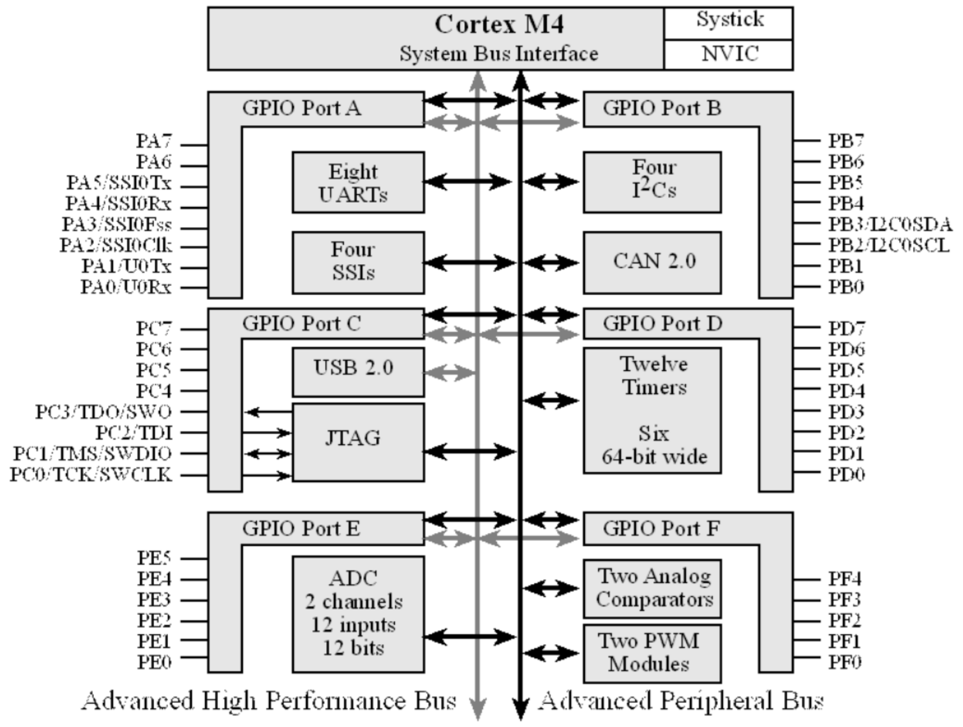
What is a microprocessor?



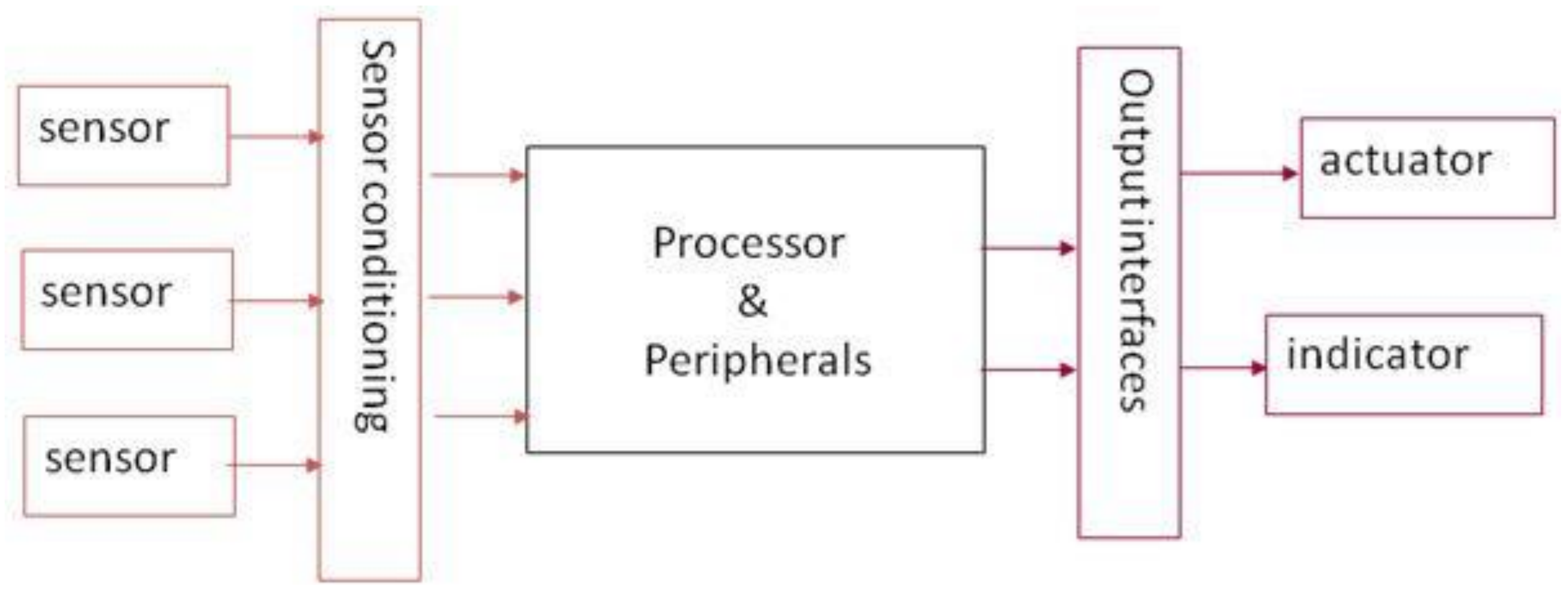
What is a microcontroller?



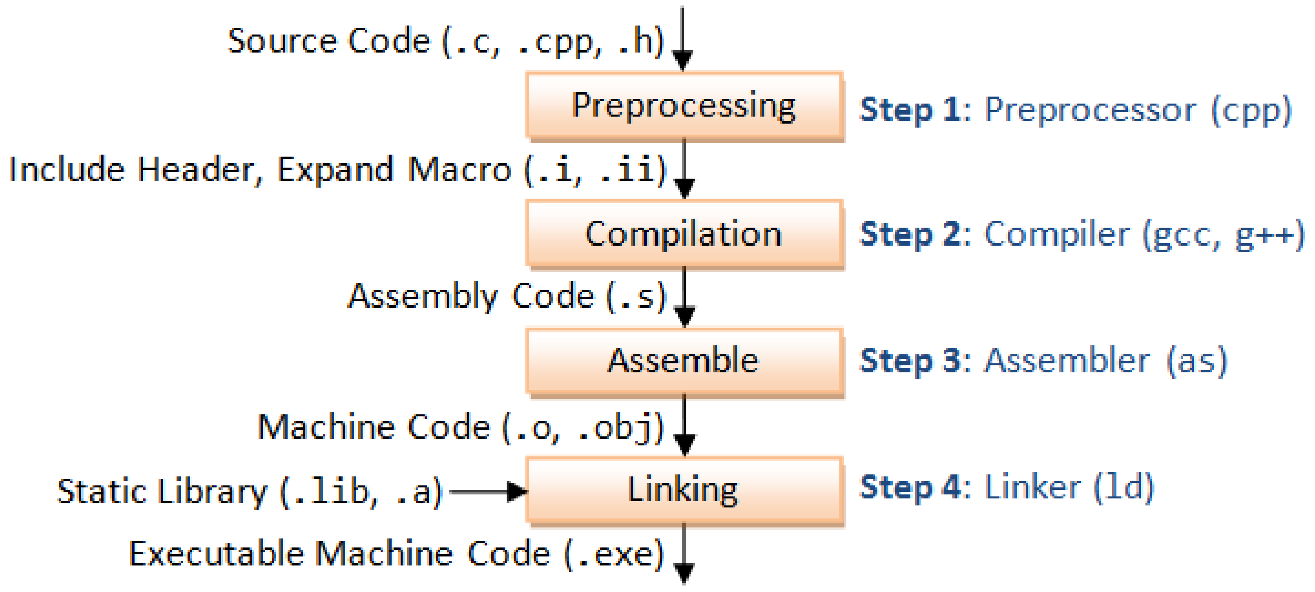
What is a microcontroller?



Embedded System



Software Build Process


Source Code (.c, .cpp, .h)
Preprocessing
Step 1: Preprocessor (cpp)
Include Header, Expand Macro (.i, .ii)
Compilation
Step 2: Compiler (gcc, g++)
Assembly Code (.s)
Assemble
Step 3: Assembler (as)
Machine Code (.o, .obj)
Static Library (.lib, .a)
Linking
Step 4: Linker (ld)
Executable Machine Code (.exe)

Introduction to Embedded C

Review of C programming concepts:

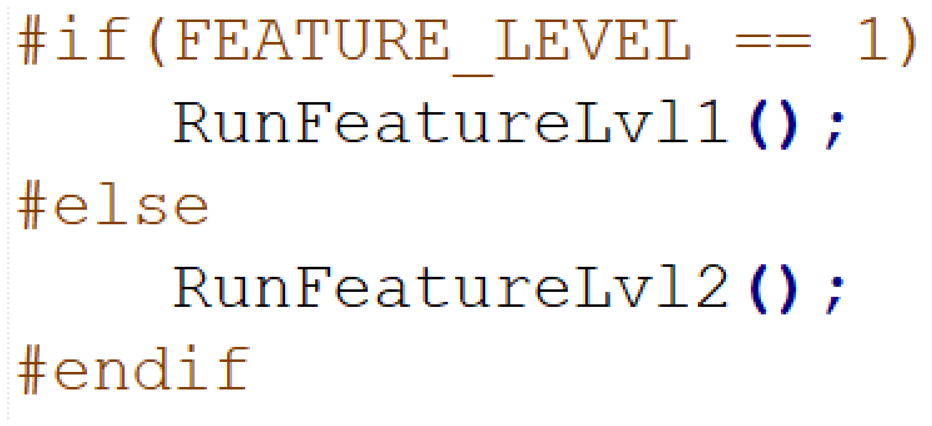
- Preprocessor Directives
- Error Types
- Primitive Data Types
- Type Casting
- Enumeration
- Structures
- Arrays
- Pointers
- Scope and Lifetime of Variables

Preprocessor Directives

- Including files
  - #include “file.h”
- Object-like macro:
  - #define WHEEL_RADIUS 10
- Function-like macro:
  - #define MAX(a,b) (((a)>(b)) ? (a):(b))
- Compiler Instructions
  - #pragma

Preprocessor Directives

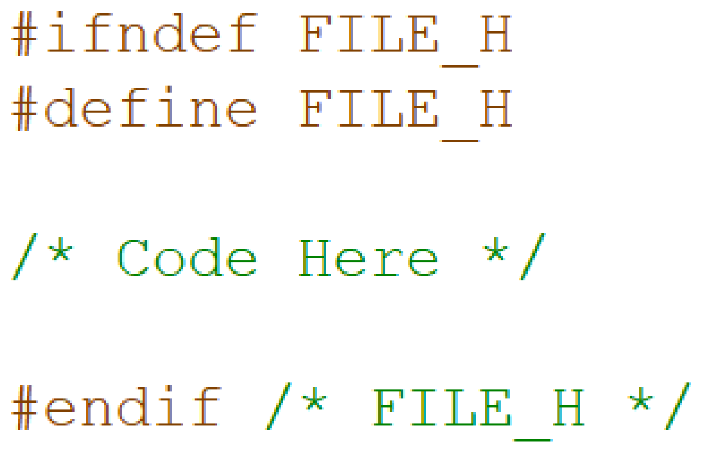
- Conditional compilation:



```
#if(FEATURE_LEVEL == 1)
    RunFeatureLvl1();
#else
    RunFeatureLvl2();
#endif
```

Preprocessor Directives

- Header file guard:

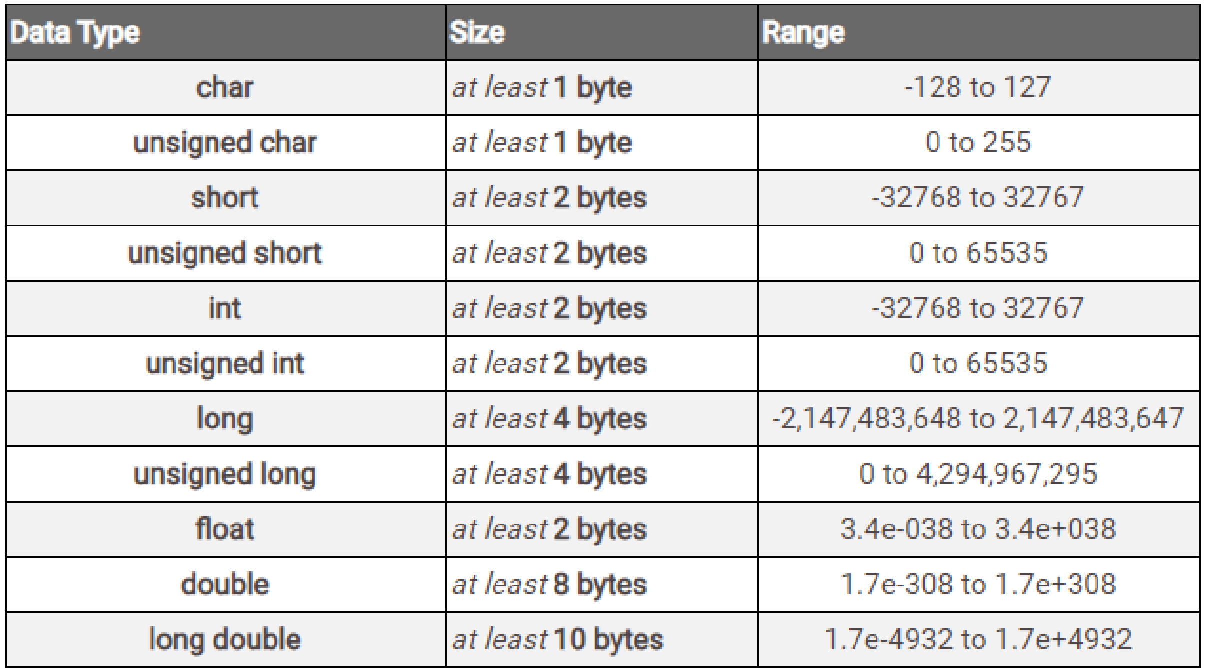


```
#ifndef FILE_H
#define FILE_H

/* Code Here */

#endif /* FILE_H */
```

Error Types

- Preproccesor Error
- Compilation Error
- Linking Error
- Logic Error

Primitive Data Types



Data Type	Size	Range
char	*at least* 1 byte	-128 to 127
unsigned char	*at least* 1 byte	0 to 255
short	*at least* 2 bytes	-32768 to 32767
unsigned short	*at least* 2 bytes	0 to 65535
int	*at least* 2 bytes	-32768 to 32767
unsigned int	*at least* 2 bytes	0 to 65535
long	*at least* 4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	*at least* 4 bytes	0 to 4,294,967,295
float	*at least* 2 bytes	3.4e-038 to 3.4e+038
double	*at least* 8 bytes	1.7e-308 to 1.7e+308
long double	*at least* 10 bytes	1.7e-4932 to 1.7e+4932

Primitive Data Types

- Standard Integer types can be included from <stdint.h>
  - int8_t
  - uint8_t
  - int16_t
  - uint16_t
  - int32_t
  - uint32_t

Type Casting

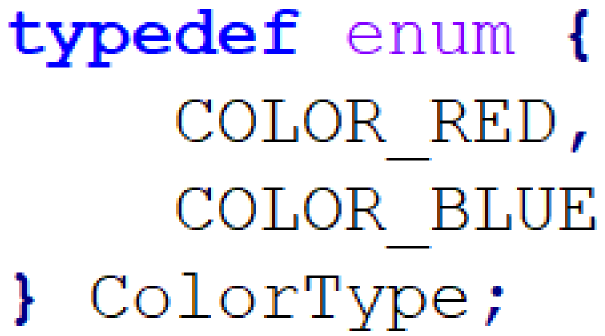
- Implicit Casting Example (should be avoided)

```
uint16_t x = 50;
// If x was bigger than 255 then truncation will occur
uint8_t  y = x;
uint32_t z = x;
```

- Explicit Casting Example

```
uint16_t x = 50;
uint8_t  y = (uint8_t) x;
uint32_t z = (uint32_t) x;
```

Enumeration

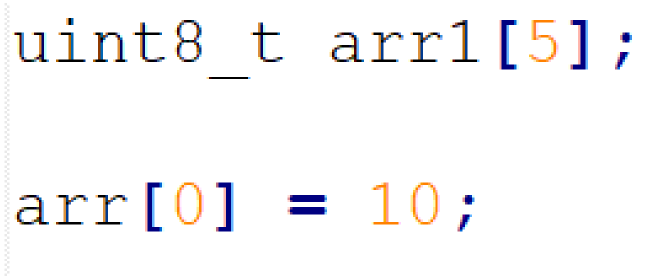


```
typedef enum {
    COLOR_RED,
    COLOR_BLUE
} ColorType;
```

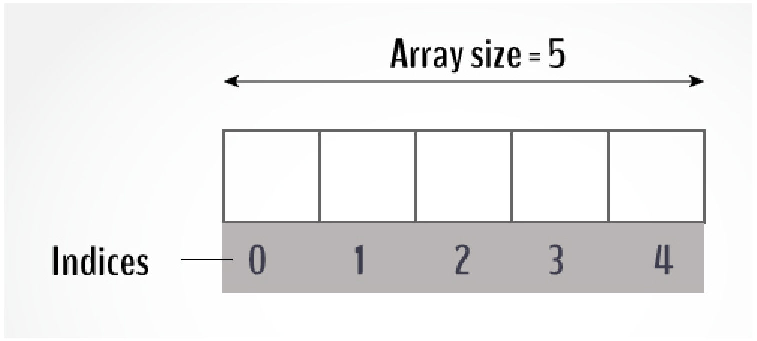
Structures

```
typedef struct {
    uint8_t id;
    uint16_t totalMarks;
} StudentType;
```

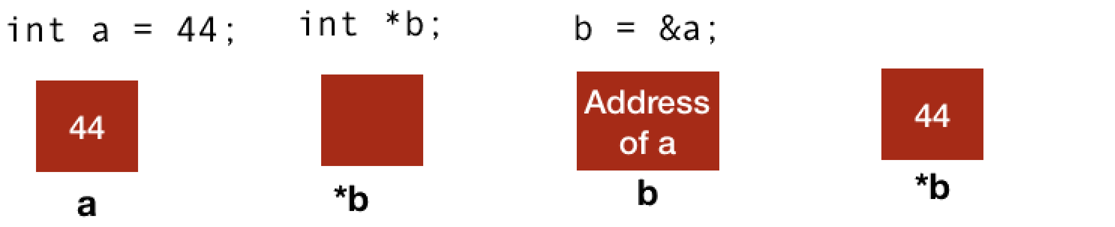
Arrays



```
uint8_t arr1[5];

arr[0] = 10;
```



Pointers



b is pointer to an integer.

b is pointing to a or
b stores the address of a

*b is value at
b (address of a)

Scope and Lifetime of Variables

- Scope:
  - **Local:** Local variable can only be accessed in the code block where it is defined
  - **File:** Global variable declared/defined with static keyword. It can only be accessed in the file where it is declared/defined.
  - **Global:** Global variable that can be accessed from all files of the project. Only one file must define the variable while other files just declare it using extern keyword

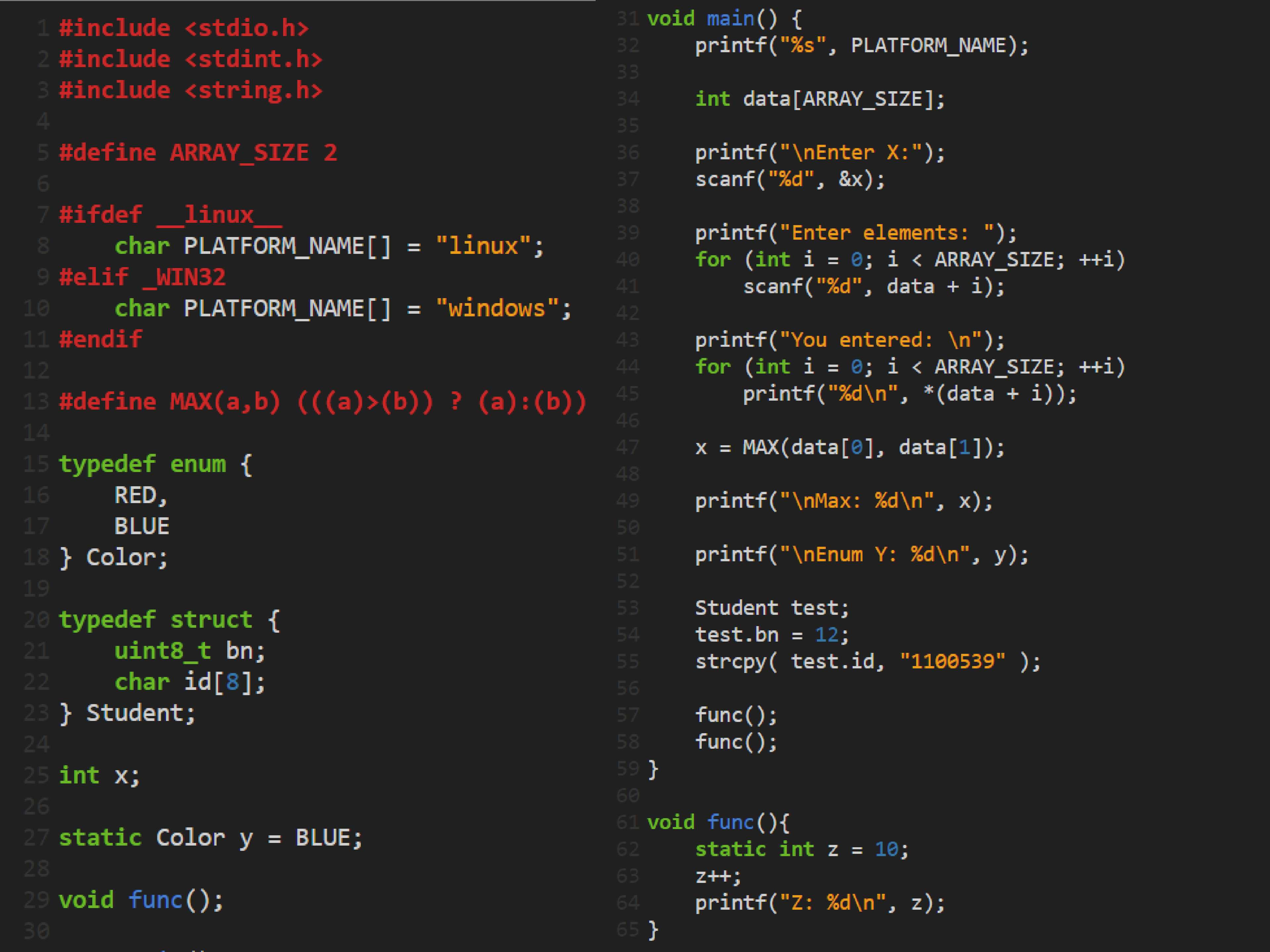
Scope and Lifetime of Variables

- Lifetime:
  - **Automatic:** lifetime ends when the block where the variable is defined ends. Automatic variables are stored in the stack.
  - **Static:** program lifetime. Static variables are stored in data memory.

Scope and Lifetime of Variables

- Example on scope and lifetime

```
uint8 globalVar = 0;                 /* This variable has global scope and static lifetime */
static uint8 fileVar = 1;            /* This variable has file scope and static lifetime */

void function(void)
{
    uint8 localVar = 2;                  /* This variable has local scope and automatic lifetime */
    static uint8 staticLocalVar = 3;     /* This variable has local scope and static lifetime */
}
```



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

#define ARRAY_SIZE 2

#ifdef __linux__
    char PLATFORM_NAME[] = "linux";
#elif _WIN32
    char PLATFORM_NAME[] = "windows";
#endif

#define MAX(a,b) (((a)>(b)) ? (a):(b))

typedef enum {
    RED,
    BLUE
} Color;

typedef struct {
    uint8_t bn;
    char id[8];
} Student;

int x;

static Color y = BLUE;

void func();

void main() {
    printf("%s", PLATFORM_NAME);

    int data[ARRAY_SIZE];

    printf("\nEnter X:");
    scanf("%d", &x);

    printf("Enter elements: ");
    for (int i = 0; i < ARRAY_SIZE; ++i)
        scanf("%d", data + i);

    printf("You entered: \n");
    for (int i = 0; i < ARRAY_SIZE; ++i)
        printf("%d\n", *(data + i));

    x = MAX(data[0], data[1]);

    printf("\nMax: %d\n", x);

    printf("\nEnum Y: %d\n", y);

    Student test;
    test.bn = 12;
    strcpy( test.id, "1100539" );

    func();
    func();
}

void func(){
    static int z = 10;
    z++;
    printf("Z: %d\n", z);
}
```