School of Mechanical Engineering
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

MECH5170M
Connected and Autonomous Vehicles Systems

MISRA C - programing standard

Kris Kubiak (k.kubiak@leeds.ac.uk)

MISRA C 2012

What is Misra C

Motor Industries Software Reliability Association

This Standard originally developed for the Automotive Industry

It produces safe and robust C.

MISRA C includes 141 rules. 121 of these are required and the remaining 20 are advisory. All rules apply to the source code and not to the object code generated by the compiler.

MISRA C 2012 Categories

Environment 4,1	Expression 9, 4
Language Extensions 3,1	Control Statement 6, 1
Documentation 5,1	Control Flow 10,0
Character Sets 2, 0	Switch Statements 5, 0
Identifiers 4,3	Functions 9, 1
Types 4, 1	Pointers and Arrays 5,1
Constants1,0	Structures and unions 4,0
Declarations and definitions 12,0	Preprocessing 13,4
Initialization 3, 0	Standard libraries 12, 0
Type conversion 6,0	Pointer type conversion3,2

MISRA C

In the group **‘Identifiers’**, a limit is defined of max 31- character significance, and the use of identical identifiers is discouraged.

In the group **‘Conversions’**, the use of implicit type conversions as well as redundant explicit casts are prohibited.

MISRA C

In the group **‘Expressions’**, a rule describes that the value of an expression should be the same under any permissible order of evaluation, and **floating-point** variables are not to be tested for exact **equality or inequality**.

In the group **‘Control Flow’**, the use of **goto, break and continue** is prohibited. Also a number of constraints on the use of the if, else, switch, and case constructs is defined.

MISRA C

The group **‘Pointers and Arrays’** prohibits the use of non-constant pointers to functions and discourages the use of pointer arithmetic at all.

The group **‘Structures and Unions’** requires that all structure/ union members are named and referred to by name only.

Rules of Misra C

Rule 3

- Assembly code and C should not be mixed.
- Real time behavior, size and other issues may require the use of assembly code.
- In this case, the mixing of the codes should be via a well defined interface.

MISRA C

RULE 6

- **Character and string** literal shall only contain that **map to the subset of ISO 10646**. Because characters are not portable between implementations.

Rule 9

- Nested Comments should flag as an error

Example

```
/*Comment
Perform_critical_thing(X);
/* Safe functionality */
```

MISRA C

Rule 13 and 17

- In the group **‘Types’**, the basic types **char, int, short, long, float** and **double** should be replaced with typedefs indicating the specific length (e.g., SI_16 for a 16 bit signed integer) and the type char shall always be declared as either unsigned char or signed char.
- Typedefs should not be reused as other typedefs for any other purpose within the same project.

instead of **int** use: **uint8_t**, **unit16_t**, **int32_t**, **int64_t** …

Eg:

typedef int int_16a;

#define int int_16a

(both should not be declared)

MISRA C

Rule 19 violation:

- Octal Constants (other than zero) shall not be used
  - A = 111;
  - B = 101;
  - C = 011;
- **Rule 20 (required):**
  - **All object and function identifiers shall be declared before use.**

MISRA C

Rule 35 and 36

- Assignment operators shall not be used in Boolean expressions
  - if ((x = y) != 0)
- bitwise operators shall not be used in boolean expressions

MISRA C

Rule 40

- If the *sizeof operator is used on an expression, it should not contain any side effects*
- *Eg:*
- *Int x,y;*
- *Y=sizeof(x=1234);*
- *// y should be assigned the value of sizeof(i) which is an integer and it is not like 1234 is assigned to i*

MISRA C

```
MY_UCHAR uc;
MY_SHORT si;
...
uc=si;
```

rule 43 violation:

- Don’t use implicit conversions which may result in information loss

MISRA C

Float EF = 10.5;

if (EF==0)

MISRA C rule 50 violation:

- Floating Point variables shall not be tested for exact equality or inequality

MISRA C

Rule 63

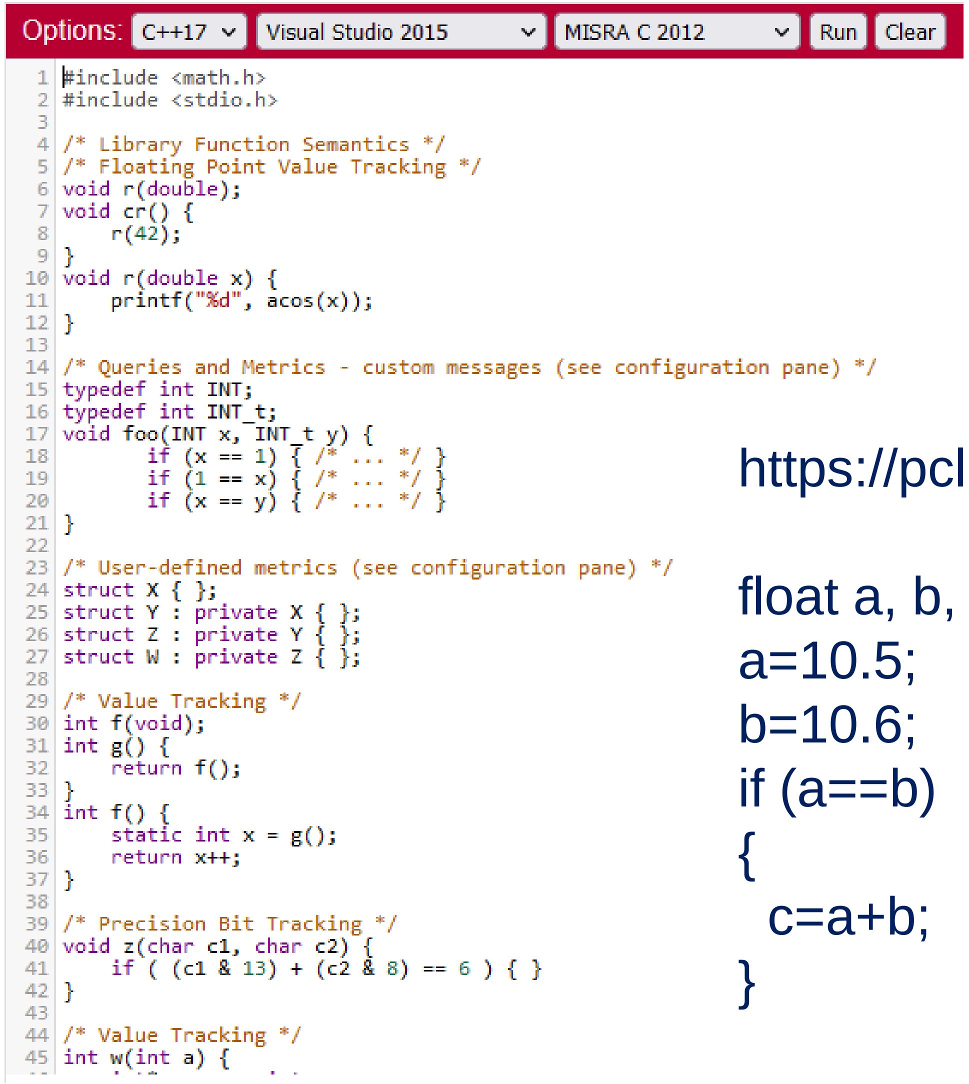
- The switch statement should not be used for only two cases, in that case if else should be used

for (F=0.0; F<10.0; F++)

rule 65 violation:

- Floating Point variables shall not be used as loop counters

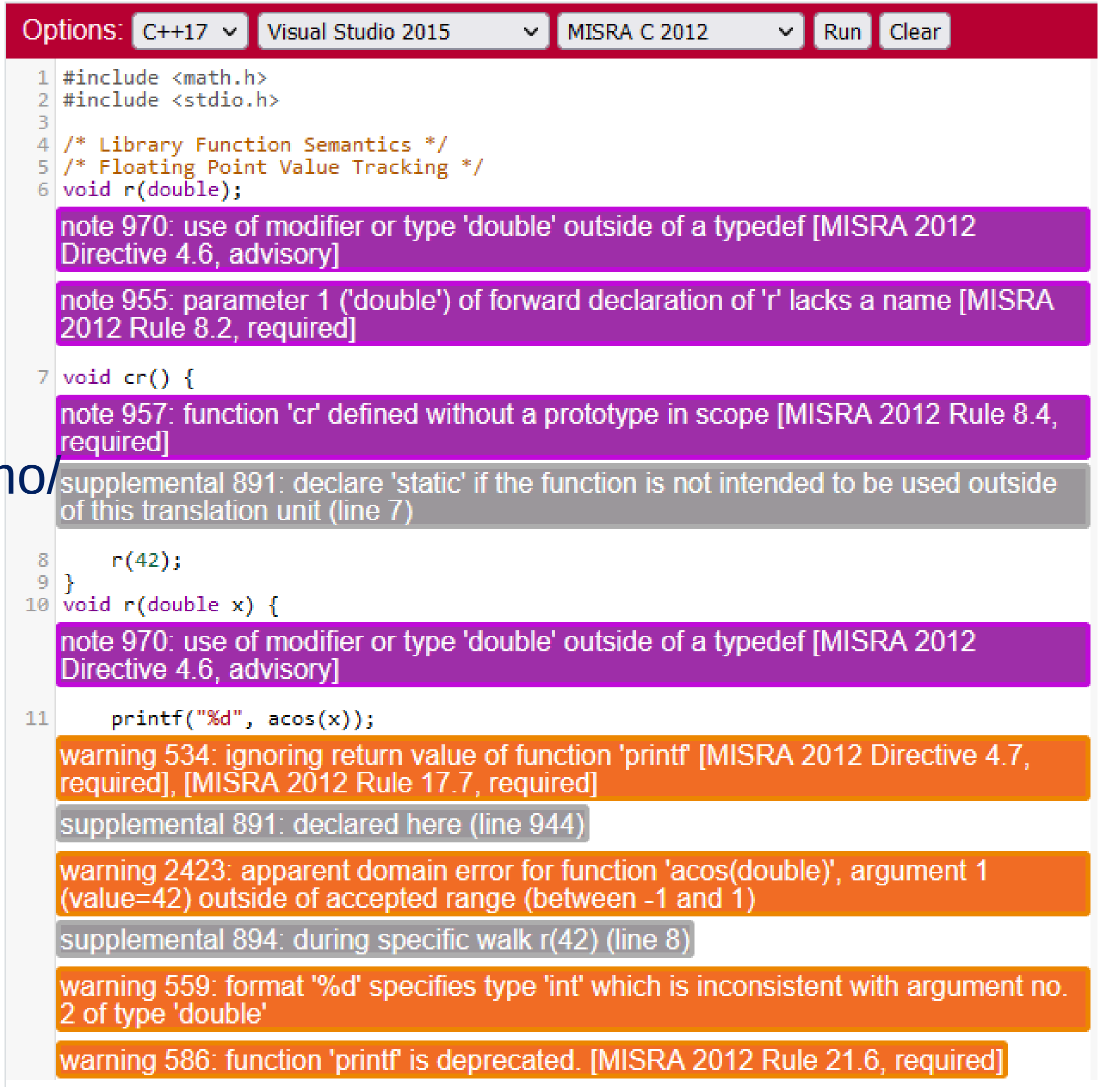
MISRA C Check



```
 1 #include <math.h>
 2 #include <stdio.h>
 3
 4 /* Library Function Semantics */
 5 /* Floating Point Value Tracking */
 6 void r(double);
 7 void cr() {
 8     r(42);
 9 }
10 void r(double x) {
11     printf("%d", acos(x));
12 }
13
14 /* Queries and Metrics - custom messages (see configuration pane) */
15 typedef int INT;
16 typedef int INT_t;
17 void foo(INT x, INT_t y) {
18         if (x == 1) { /* ... */ }
19         if (1 == x) { /* ... */ }
20         if (x == y) { /* ... */ }
21 }
22
23 /* User-defined metrics (see configuration pane) */
24 struct X { };
25 struct Y : private X { };
26 struct Z : private Y { };
27 struct W : private Z { };
28
29 /* Value Tracking */
30 int f(void);
31 int g() {
32     return f();
33 }
34 int f() {
35     static int x = g();
36     return x++;
37 }
38
39 /* Precision Bit Tracking */
40 void z(char c1, char c2) {
41     if ( (c1 & 13) + (c2 & 8) == 6 ) { }
42 }
43
44 /* Value Tracking */
45 int w(int a) {
```

https://pclintplus.com/demo/

float a, b, c;
a=10.5;
b=10.6;
if (a==b)
{
  c=a+b;
}



```
 1 #include <math.h>
 2 #include <stdio.h>
 3
 4 /* Library Function Semantics */
 5 /* Floating Point Value Tracking */
 6 void r(double);
```

note 970: use of modifier or type 'double' outside of a typedef [MISRA 2012 Directive 4.6, advisory]

note 955: parameter 1 ('double') of forward declaration of 'r' lacks a name [MISRA 2012 Rule 8.2, required]

```
 7 void cr() {
```

note 957: function 'cr' defined without a prototype in scope [MISRA 2012 Rule 8.4, required]

supplemental 891: declare 'static' if the function is not intended to be used outside of this translation unit (line 7)

```
 8     r(42);
 9 }
10 void r(double x) {
```

note 970: use of modifier or type 'double' outside of a typedef [MISRA 2012 Directive 4.6, advisory]

```
11     printf("%d", acos(x));
```

warning 534: ignoring return value of function 'printf' [MISRA 2012 Directive 4.7, required], [MISRA 2012 Rule 17.7, required]

supplemental 891: declared here (line 944)

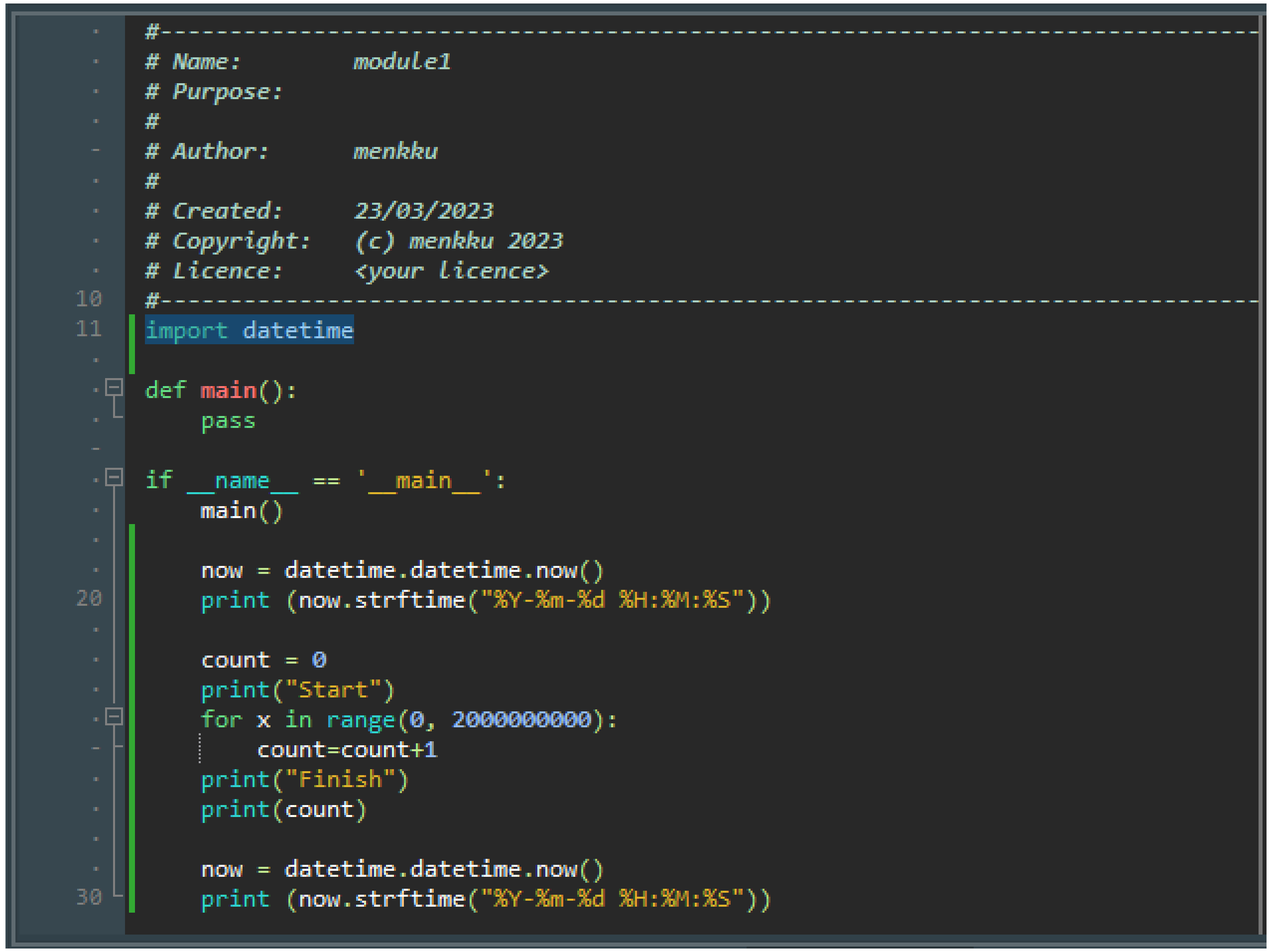
warning 2423: apparent domain error for function 'acos(double)', argument 1 (value=42) outside of accepted range (between -1 and 1)

supplemental 894: during specific walk r(42) (line 8)

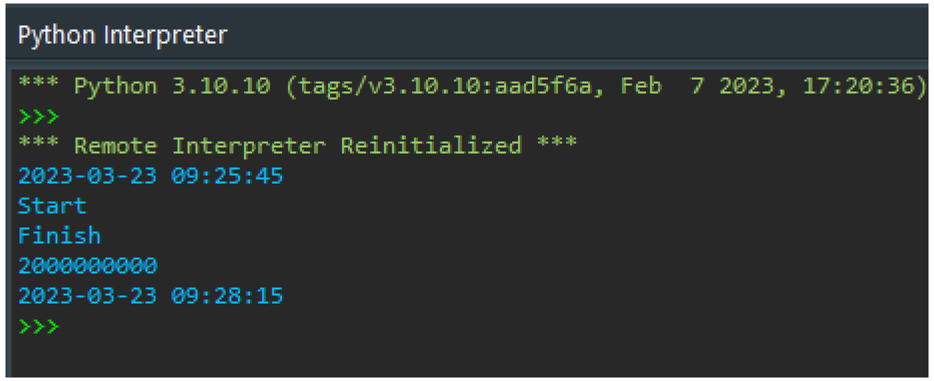
warning 559: format '%d' specifies type 'int' which is inconsistent with argument no. 2 of type 'double'

warning 586: function 'printf' is deprecated. [MISRA 2012 Rule 21.6, required]

Python Speed



```
#-------------------------------------------------------------------------------
# Name:        module1
# Purpose:
#
# Author:      menkku
#
# Created:     23/03/2023
# Copyright:   (c) menkku 2023
# Licence:     <your licence>
#-------------------------------------------------------------------------------
import datetime

def main():
    pass

if __name__ == '__main__':
    main()

    now = datetime.datetime.now()
    print (now.strftime("%Y-%m-%d %H:%M:%S"))

    count = 0
    print("Start")
    for x in range(0, 2000000000):
        count=count+1
    print("Finish")
    print(count)

    now = datetime.datetime.now()
    print (now.strftime("%Y-%m-%d %H:%M:%S"))
```

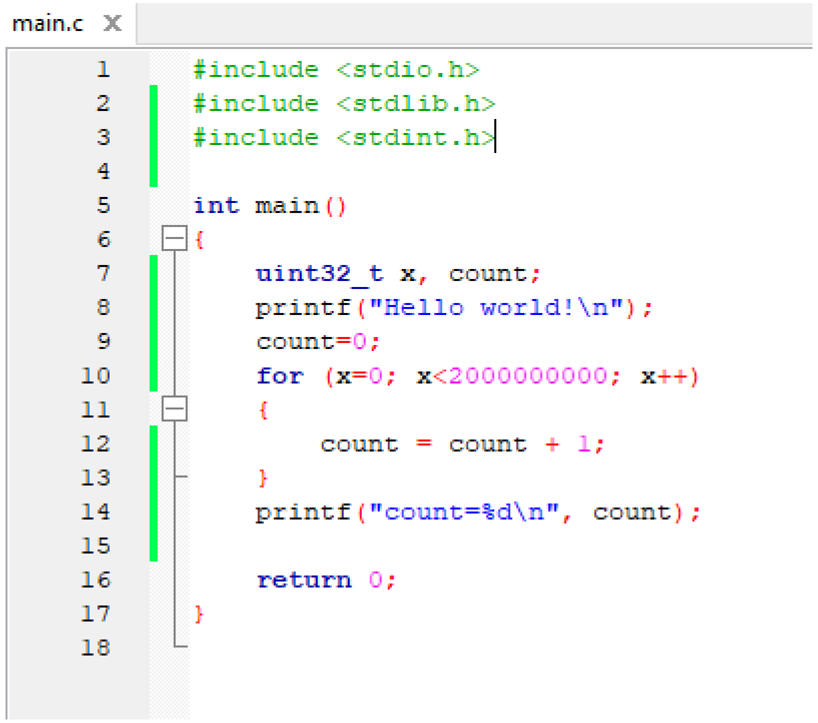


Python Interpreter

```
*** Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36)
>>>
*** Remote Interpreter Reinitialized ***
2023-03-23 09:25:45
Start
Finish
2000000000
2023-03-23 09:28:15
>>>
```

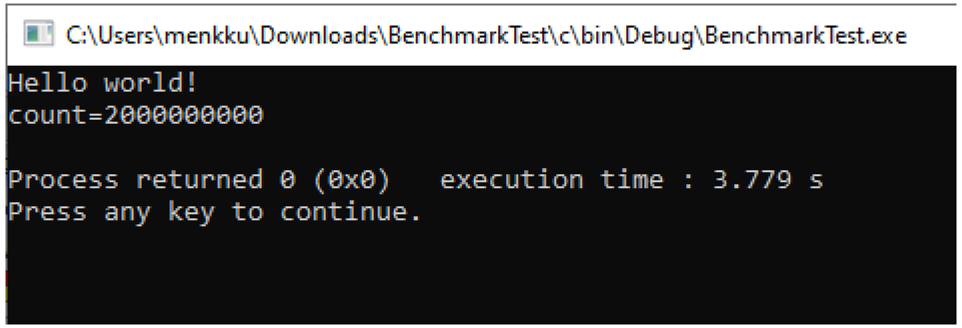
Execution time 2min 30s

C99 Speed



main.c

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

int main()
{
    uint32_t x, count;
    printf("Hello world!\n");
    count=0;
    for (x=0; x<2000000000; x++)
    {
        count = count + 1;
    }
    printf("count=%d\n", count);

    return 0;
}
```

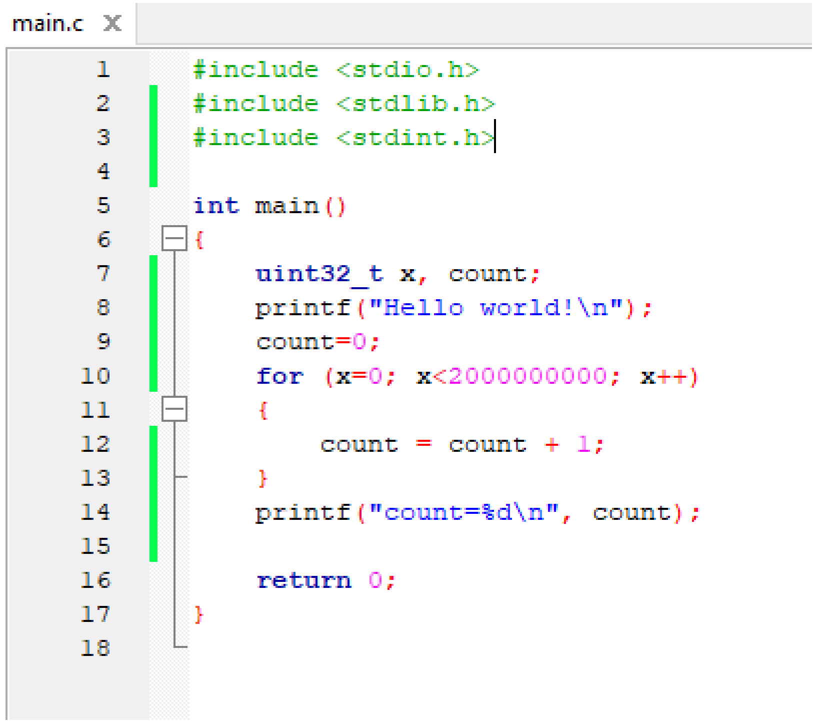


C:\Users\menkku\Downloads\BenchmarkTest\c\bin\Debug\BenchmarkTest.exe

```
Hello world!
count=2000000000

Process returned 0 (0x0)   execution time : 3.779 s
Press any key to continue.
```

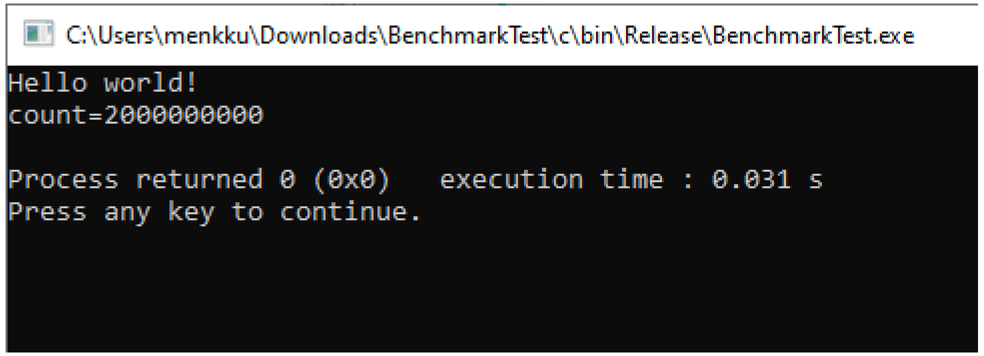
Execution time 3.779s

C99 Speed



main.c

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

int main()
{
    uint32_t x, count;
    printf("Hello world!\n");
    count=0;
    for (x=0; x<2000000000; x++)
    {
        count = count + 1;
    }
    printf("count=%d\n", count);

    return 0;
}
```



C:\Users\menkku\Downloads\BenchmarkTest\c\bin\Release\BenchmarkTest.exe

```
Hello world!
count=2000000000

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

Execution time 0.031s

Conclusions

- MISRA C is a source code standard
- Used for safety critical systems
- Eliminates common mistakes and errors
- Industry standard for embedded systems
- Python, Matlab are interpreted languages (slow)
- C is compiled to a machine code (fast)

Q&A

UNIVERSITY OF LEEDS

ANY QUESTIONS

???