

C Program: Leap Year

The purpose of this program is to determine if a given year is a leap by using mathematical logic

Original code from:

<https://www.sanfoundry.com/c-program-check-year-leap/>

```
1. /*
2.  * C program to find whether a given year is leap year or not
3.  */
4. void main()
5. {
6.     int year;
7.
8.     printf("Enter a year \n");
9.     scanf("%d", &year);
10.    if ((year % 400) == 0)
11.        printf("%d is a leap year \n", year);
12.    else if ((year % 100) == 0)
13.        printf("%d is a not leap year \n", year);
14.    else if ((year % 4) == 0)
15.        printf("%d is a leap year \n", year);
16.    else
17.        printf("%d is not a leap year \n", year);
18. }
```

The code from *sanfoundry.com* would not compile right so I changed some operators/operands in the code and modified the `if-else` statement to produce a cleaner and easier to digest code, user preference of course. Moreover, I added a greeting (`Welcome!`) with instructions in the beginning of the code to greet and instruct user. Further I added some line advances (`\n`)

Code outputs:

```
ASM generation compiler returned: 0
Execution build compiler returned: 0
Program returned: 0
Welcome!
To find out if a given year is a leap year, please enter a year below.

Enter year:
0 is a Leap Year
```

Modified code used for this exercise:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int y;
6
7      printf("Welcome!\n");
8      printf("To find out if a given year is a leap year, please enter a year below.\n");
9
10     printf("\nEnter year: \n");
11     scanf("%d",&y);
12
13     if(y % 4 == 0)
14     {
15         if( y % 100 == 0)
16         {
17             if( y % 400 == 0)
18                 printf("\n%d is a Leap Year", y);
19             else
20                 printf("\n%d is not a Leap Year", y);
21         }
22         else
23             printf("%d is a Leap Year", y );
24     }
25     else
26         printf("%d is not a Leap Year", y);
27
28     return 0;
29 }
```

Code Assembly Equivalent:

x86-64 gcc 9.1 (C, Editor #1, Compiler #1)

x86-64 gcc 9.1

Compiler options...

Output...

Filter...

Libraries

+ Add new...

Add tool...

```
1  .LC0:
2      .string "Welcome!"
3  .LC1:
4      .string "To find out if a given year is a leap year, please enter a year below."
5  .LC2:
6      .string "\nEnter year: "
7  .LC3:
8      .string "%d"
9  .LC4:
10     .string "\n%d is a Leap Year"
11 .LC5:
12     .string "\n%d is not a Leap Year"
13 .LC6:
14     .string "%d is a Leap Year"
15 .LC7:
16     .string "%d is not a Leap Year"
17 main:
18     push    rbp
19     mov     rbp, rsp
20     sub     rsp, 16
21     mov     edi, OFFSET FLAT:.LC0
22     call    puts
23     mov     edi, OFFSET FLAT:.LC1
24     call    puts
25     mov     edi, OFFSET FLAT:.LC2
26     call    puts
27     lea     rax, [rbp-4]
28     mov     rsi, rax
29     mov     edi, OFFSET FLAT:.LC3
30     mov     eax, 0
31     call    __isoc99_scanf
32     mov     eax, DWORD PTR [rbp-4]
33     and     eax, 3
34     test    eax, eax
35     jne     .L2
36     mov     edx, DWORD PTR [rbp-4]
37     movsx   rax, edx
38     imul    rax, rax, 1374389535
39     shr     rax, 32
40     mov     ecx, eax
41     sar     ecx, 5
42     mov     eax, edx
43     sar     eax, 31
44     sub     ecx, eax
45     mov     eax, ecx
46     imul    eax, eax, 100
47     sub     edx, eax
48     mov     eax, edx
49     test    eax, eax
50     jne     .L3
51     mov     edx, DWORD PTR [rbp-4]
52     movsx   rax, edx
53     imul    rax, rax, 1374389535
```

Output (0/0)

x86-64 gcc 9.1

- cached (12556B) ~790 lines filtered

Code Hexadecimal Equivalent:

x86-64 gcc 9.1 (C, Editor #1, Compiler #1)

x86-64 gcc 9.1 Compiler options...

A Output... Filter... Libraries + Add new... Add tool...

```
printf@plt:
_dl_relocate_static_pie:
main:
55
4005b2 push    rbp
48 89 e5
4005b3 mov     rbp, rsp
48 83 ec 10
4005b6 sub     rsp, 0x10
bf 40 07 40 00
4005ba mov     edi, 0x400740
e8 dc fe ff ff
4005bf call    4004a0 <puts@plt>
bf 50 07 40 00
4005c4 mov     edi, 0x400750
e8 d2 fe ff ff
4005c9 call    4004a0 <puts@plt>
bf 97 07 40 00
4005ce mov     edi, 0x400797
e8 c8 fe ff ff
4005d3 call    4004a0 <puts@plt>
48 8d 45 fc
4005d8 lea     rax, [rbp-0x4]
48 89 c6
4005dc mov     rsi, rax
bf a5 07 40 00
4005df mov     edi, 0x4007a5
b8 00 00 00 00
4005e4 mov     eax, 0x0
e8 d2 fe ff ff
4005e9 call    4004c0 <__isoc99_scanf@plt>
8b 45 fc
4005ee mov     eax, DWORD PTR [rbp-0x4]
83 e0 03
4005f1 and     eax, 0x3
85 c0
4005f4 test    eax, eax
0f 85 99 00 00 00
4005f6 jne     400695 <main+0xe3>
8b 55 fc
4005fc mov     edx, DWORD PTR [rbp-0x4]
48 63 c2
4005ff movsxd  rax, edx
48 69 c0 1f 85 eb 51
400602 imul    rax, rax, 0x51eb851f
48 c1 e8 20
400609 shr     rax, 0x20
89 c1
40060d mov     ecx, eax
c1 f9 05
40060f sar     ecx, 0x5
89 d0
400612 mov     eax, edx
```

Output (0/0) x86-64 gcc 9.1 - cached (108008) ~184 lines filtered

Analyzing One Line of Code:

10

```
printf("\nEnter year: \n");
```

This line of code `printf("\nEnter year: \n");` is executing two things. `mov` It's moving data back and from a hardware-supported stack in RAM into CPU registers. Additionally, `call` calls near procedures, causes the procedure named in the operand to be executed.

According to *godbolt.org* compiler:

Copies the second operand (source operand) to the first operand (destination operand). The source operand can be an immediate value, general-purpose register, segment register, or memory location; the destination register can be a general-purpose register, segment register, or memory location.

25

```
mov edi, OFFSET FLAT:LC2
```

26

```
call puts
```

According to *godbolt.org* compiler:

Saves procedure linking information on the stack and branches to the called procedure specified using the target operand. The target operand specifies the address of the first instruction in the called procedure. The operand can be an immediate value, a general-purpose register, or a memory location.