




(/)



Curriculum

SE Foundations Average: 108.76% 

0x03. C - Debugging

C**Debugging** By: Carrie Ybay Weight: 1 Project over - took place from Feb 21, 2023 6:00 AM to Feb 24, 2023 6:00 AM☒ An auto review will be launched at the deadline

In a nutshell...

- **Auto QA review:** 40.0/40 mandatory
- **Altogether: 100.0%**
 - Mandatory: 100.0%
 - Optional: no optional tasks

Resources

Read or watch:

- Debugging (/rltoken/faGcpiJiejHH6GhqpmhUw)
- Rubber Duck Debugging (/rltoken/RaecqJBNkmZ92vLMpNDuGg)

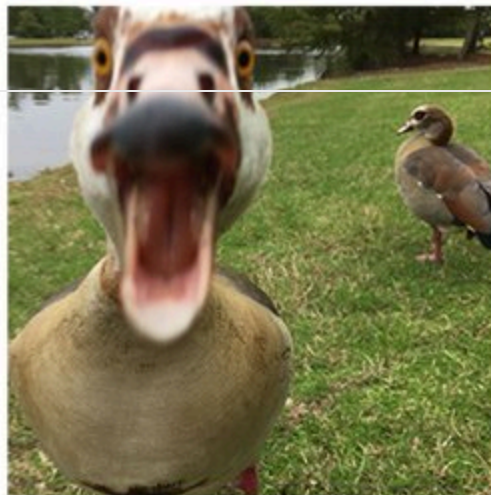
Debugging is the process of finding and fixing errors in software that prevents it from running correctly. As you become a more advanced programmer and an industry engineer, you will learn how to use debugging tools such as `gdb` or built-in tools that IDEs have. However, it's important to understand the concepts and processes of debugging manually.



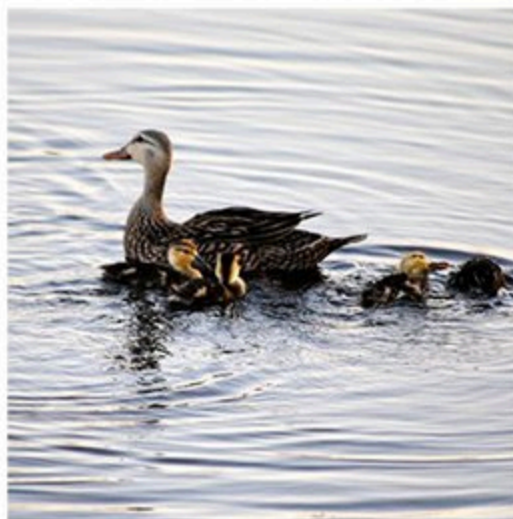
Help

(/)

He attac

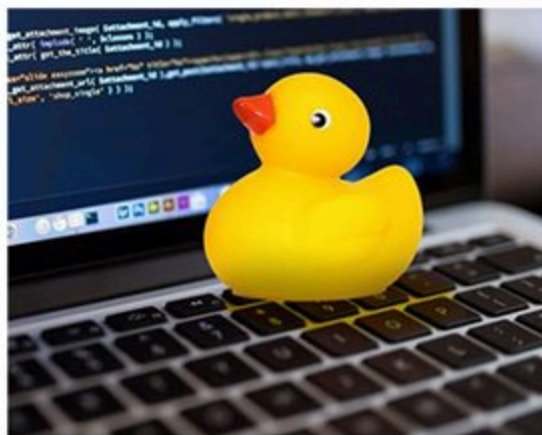


He protec



but most importantly

he prevent overflow on stac



Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/b8uX1nly0A55tWVIIRTaHQ), without the help of Google:

General

- What is debugging



- What are some methods of debugging manually
- (/).• How to read the error messages

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- Your code should use the `Betty` style. It will be checked using `betty-style.pl` and `betty-doc.pl`
- A `README.md` file at the root of the repo, containing a description of the repository
- A `README.md` file, at the root of the folder of this project (i.e. `0x03-debugging`), describing what this project is about

Quiz questions

Great! You've completed the quiz successfully! Keep going! ([Show quiz](#))

Tasks

0. Multiple mains

mandatory

Score: 100.0% (Checks completed: 100.0%)

In most projects, we often give you only one main file to test with. For example, this main file is a test for a `positive_or_negative()` function similar to the one you worked with in an earlier C project ([/rltoken/IKcOFkG-GCivSDXgWgld2g](#)):



```
carrie@ubuntu:/debugging$ cat main.c
#include "main.h"

/**
 * main - tests function that prints if integer is positive or negative
 * Return: 0
 */

int main(void)
{
    int i;

    i = 98;
    positive_or_negative(i);

    return (0);
}
carrie@ubuntu:/debugging$
```

```
carrie@ubuntu:/debugging$ cat main.h
#ifndef MAIN_H
#define MAIN_H

#include <stdio.h>

void positive_or_negative(int i);

#endif /* MAIN_H */
carrie@ubuntu:/debugging$
```

```
carrie@ubuntu:/debugging$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 positive_or_negative.c main.c
carrie@ubuntu:/debugging$ ./a.out
98 is positive
carrie@ubuntu:/debugging$
```

Based on the `main.c` file above, create a file named `0-main.c`. This file must test that the function `positive_or_negative()` gives the correct output when given a case of `0`.

You are not coding the solution / function, you're just testing it! However, you can adapt your function from 0x01. C - Variables, if, else, while - Task #0 (/rltoken/IKcOFkG-GCivSDXgWgld2g) to compile with this main file to test locally.

- You only need to upload `0-main.c` and `main.h` for this task. We will provide our own `positive_or_negative()` function.
- You are not allowed to add or remove lines of code, you may change only **one** line in this task.



```
carrie@ubuntu:/debugging$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 positive_or  
_negative.c 0-main.c -o 0-main  
carrie@ubuntu:/debugging$ ./0-main  
0 is zero  
carrie@ubuntu:/debugging$ wc -l 0-main.c  
16 1-main.c  
carrie@ubuntu:/debugging$
```

Repo:

- GitHub repository: alx-low_level_programming
- Directory: 0x03-debugging
- File: 0-main.c, main.h

☒ Done![Help](#)[Check your code](#)[QA Review](#)**1. Like, comment, subscribe****mandatory**Score: 100.0% (*Checks completed: 100.0%*)

Copy this main file. Comment out (don't delete it!) the part of the code that is causing the output to go into an infinite loop.

- Don't add or remove any lines of code, as we will be checking your line count. You are only allowed to comment out existing code.
- You do not have to compile with `-Wall -Werror -Wextra -pedantic` for this task.



```
carrie@ubuntu:/debugging$ cat 1-main.c
#include <stdio.h>

/**
 * main - causes an infinite loop
 * Return: 0
 */

int main(void)
{
    int i;

    printf("Infinite loop incoming :(\n");

    i = 0;

    while (i < 10)
    {
        putchar(i);
    }

    printf("Infinite loop avoided! \o/\n");

    return (0);
}
carrie@ubuntu:/debugging$
```

Your output should look like this:

```
carrie@ubuntu:/debugging$ gcc -std=gnu89 1-main.c -o 1-main
carrie@ubuntu:/debugging$ ./1-main
Infinite loop incoming :(
Infinite loop avoided! \o/
carrie@ubuntu:/debugging$ wc -l 1-main.c
24 1-main.c
carrie@ubuntu:/debugging$
```

Repo:

- GitHub repository: alx-low_level_programming
- Directory: 0x03-debugging
- File: 1-main.c

☒ Done![Help](#)[Check your code](#)[_ Get a sandbox](#)[QA Review](#)

2.0 > 972?

mandatory

Score: 100.0% (Checks completed: 100.0%)

~~(/)~~
This program prints the largest of three integers.

```
carrie@ubuntu:/debugging$ cat 2-main.c
#include <stdio.h>
#include "main.h"

/**
 * main - prints the largest of 3 integers
 * Return: 0
 */

int main(void)
{
    int a, b, c;
    int largest;

    a = 972;
    b = -98;
    c = 0;

    largest = largest_number(a, b, c);

    printf("%d is the largest number\n", largest);

    return (0);
}
carrie@ubuntu:/debugging$
```



```
carrie@ubuntu:/debugging$ cat 2-largest_number.c
```

```
#include "main.h"
```

```
/**
 * largest_number - returns the largest of 3 numbers
 * @a: first integer
 * @b: second integer
 * @c: third integer
 * Return: largest number
 */
```

```
int largest_number(int a, int b, int c)
{
    int largest;

    if (a > b && b > c)
    {
        largest = a;
    }
    else if (b > a && a > c)
    {
        largest = b;
    }
    else
    {
        largest = c;
    }

    return (largest);
}
```

```
carrie@ubuntu:/debugging$
```

```
carrie@ubuntu:/debugging$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 2-largest_n  
umber.c 2-main.c -o 2-main
```

```
carrie@ubuntu:/debugging$ ./2-main
```

```
0 is the largest number
```

```
carrie@ubuntu:/debugging$
```

? That's definitely not right.

Fix the code in `2-largest_number.c` so that it correctly prints out the largest of three numbers, no matter the case.

- Line count will not be checked for this task.

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x03-debugging`



- File: 2-largest_number.c, main.h (/)

☒ Done!

Help

Check your code

>_ Get a sandbox

QA Review

3. Leap year

mandatory

Score: 100.0% (Checks completed: 100.0%)

This program converts a date to the day of year and determines how many days are left in the year, taking leap year into consideration.

```
carrie@ubuntu:/debugging$ cat 3-main_a.c
#include <stdio.h>
#include "main.h"

/**
 * main - takes a date and prints how many days are left in the year, taking
 * leap years into account
 * Return: 0
 */

int main(void)
{
    int month;
    int day;
    int year;

    month = 4;
    day = 01;
    year = 1997;

    printf("Date: %02d/%02d/%04d\n", month, day, year);

    day = convert_day(month, day);

    print_remaining_days(month, day, year);

    return (0);
}

carrie@ubuntu:/debugging$
```



```
carrie@ubuntu:/debugging$ cat 3-convert_day.c
#include "main.h"
```

```
/**
 * convert_day - converts day of month to day of year, without accounting
 * for leap year
 * @month: month in number format
 * @day: day of month
 * Return: day of year
 */

int convert_day(int month, int day)
{
    switch (month)
    {
        case 2:
            day = 31 + day;
            break;
        case 3:
            day = 59 + day;
            break;
        case 4:
            day = 90 + day;
            break;
        case 5:
            day = 120 + day;
            break;
        case 6:
            day = 151 + day;
            break;
        case 7:
            day = 181 + day;
            break;
        case 8:
            day = 212 + day;
            break;
        case 9:
            day = 243 + day;
            break;
        case 10:
            day = 273 + day;
            break;
        case 11:
            day = 304 + day;
            break;
        case 12:
            day = 334 + day;
            break;
        default:
            break;
    }
    return (day);
}
```



```
}  
(/)  
carrie@ubuntu:/debugging$
```

```
carrie@ubuntu:/debugging$ cat 3-print_remaining_days.c  
#include <stdio.h>  
#include "main.h"  
  
/**  
 * print_remaining_days - takes a date and prints how many days are  
 * left in the year, taking leap years into account  
 * @month: month in number format  
 * @day: day of month  
 * @year: year  
 * Return: void  
 */  
  
void print_remaining_days(int month, int day, int year)  
{  
    if ((year % 4 == 0 || year % 400 == 0) && !(year % 100 == 0))  
    {  
        if (month >= 2 && day >= 60)  
        {  
            day++;  
        }  
  
        printf("Day of the year: %d\n", day);  
        printf("Remaining days: %d\n", 366 - day);  
    }  
    else  
    {  
        if (month == 2 && day == 60)  
        {  
            printf("Invalid date: %02d/%02d/%04d\n", month, day - 31, year);  
        }  
        else  
        {  
            printf("Day of the year: %d\n", day);  
            printf("Remaining days: %d\n", 365 - day);  
        }  
    }  
}
```

```
carrie@ubuntu:/debugging$
```



```
carrie@ubuntu:/debugging$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 3-convert_d
ay.c 3-print_remaining_days.c 3-main_a.c -o 3-main_a
carrie@ubuntu:/debugging$ ./3-main_a
Date: 04/01/1997
Day of the year: 91
Remaining days: 274
carrie@ubuntu:/debugging$
```

Output looks good for 04/01/1997 ! Let's make a new main file 3-main_b.c to try a case that is a leap year: 02/29/2000 .

```
carrie@ubuntu:/debugging$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 3-convert_d
ay.c 3-print_remaining_days.c 3-main_b.c -o 3-main_b
carrie@ubuntu:/debugging$ ./3-main_b
Date: 02/29/2000
Invalid date: 02/29/2000
carrie@ubuntu:/debugging$
```

? That doesn't seem right.

Fix the `print_remaining_days()` function so that the output works correctly for *all* dates and *all* leap years.

- Line count will not be checked for this task.
- You can assume that all test cases have valid months (i.e. the value of `month` will never be less than 1 or greater than 12) and valid days (i.e. the value of `day` will never be less than 1 or greater than 31).
- You can assume that all test cases have valid month/day combinations (i.e. there will never be a June 31st or November 31st, etc.), but not all month/day/year combinations are valid (i.e. February 29, 1991 or February 29, 2427).

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x03-debugging`
- File: `3-print_remaining_days.c`, `main.h`

☒ Done![Help](#)[Check your code](#)[QA Review](#)