

# System Software

## Introduction to course

Jarno Tuominen, Spring 2024

# Course Practises

- Schedule
- Exercises
- Grading



# Schedule

Lectures mainly on Fridays (a few Wednesdays as well)

1 hour theory part (lecture)

2 hours demo/exercise session

- Occasionally teacher may (or may not) show some demonstration
- Mainly you will be working with the exercises, while teacher is supporting you on a need basis

Presence is mandatory - no online/ recorded lectures

Communications over Teams - feel free to ask help!



# Exercises

Two systems in use

- Ville for automatically graded exercises
- Virtual machine-based development environment on your own PC
- Git for submitting your work

Note: ItsLearning not in use. All the material in Git. Follow also Teams.



# Grading

Based on:

- Completion rate of exercises
- Git activity
- A short exam testing basic concepts

Weighting to be decided (later on)



# Questions?



# **System Software**

## **Basics of C Language**

Sanna Määttä

# C Language

# C Language

Originally C was designed for UNIX operating system.

C was created from ALGOL, BCPL and B programming languages and was first published in 1972.

Latest C standard is from 2018, ISO/IEC 9899:2018 (C17 is the informal name).

- The ISO/IEC standard is not free.
- You can find a final draft version of the 2011 standard here: <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf>



# C Language

C is a general purpose and procedural programming language.

C has a static type system.

- This is different compared to e.g. Python, where you do not always need to define the type of a variable.



# Where Is C Language Used?

Embedded Systems, IoT applications

In System Software (operating system kernels, device drivers, Bootloader, BIOS/UEFI, system applications...)

Most operating systems (at least their kernels) are written in C (UNIX, Linux, Windows, MacOS, iOS, Android, Windows Phone).

Most databases (Oracle DB, MySQL, PostgreSQL, MS SQL Server) are written in C/C++.

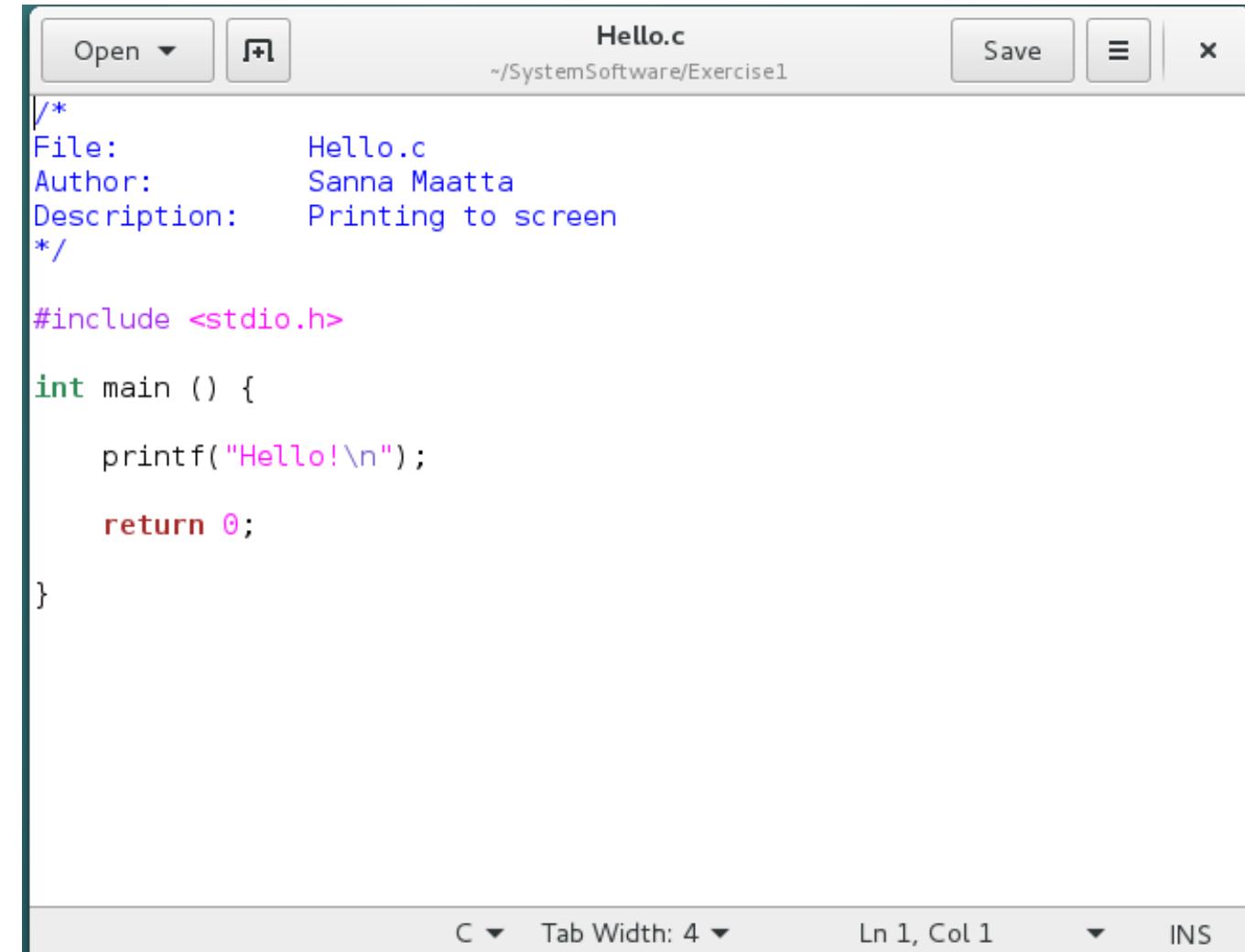


# First Program in C

# How to Start Writing Code

Write the “Hello World” program first.

Then compile and run your code, this way you know your development environment is working properly!



The screenshot shows a code editor window titled "Hello.c" located at "~/SystemSoftware/Exercise1". The window includes standard file operations like "Open", "Save", and "Close". The code itself is a simple "Hello World" program:

```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

The code editor also displays status information at the bottom: "C", "Tab Width: 4", "Ln 1, Col 1", and "INS".



Open Hello.c  
~/SystemSoftware/Exercise1 Save

```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>

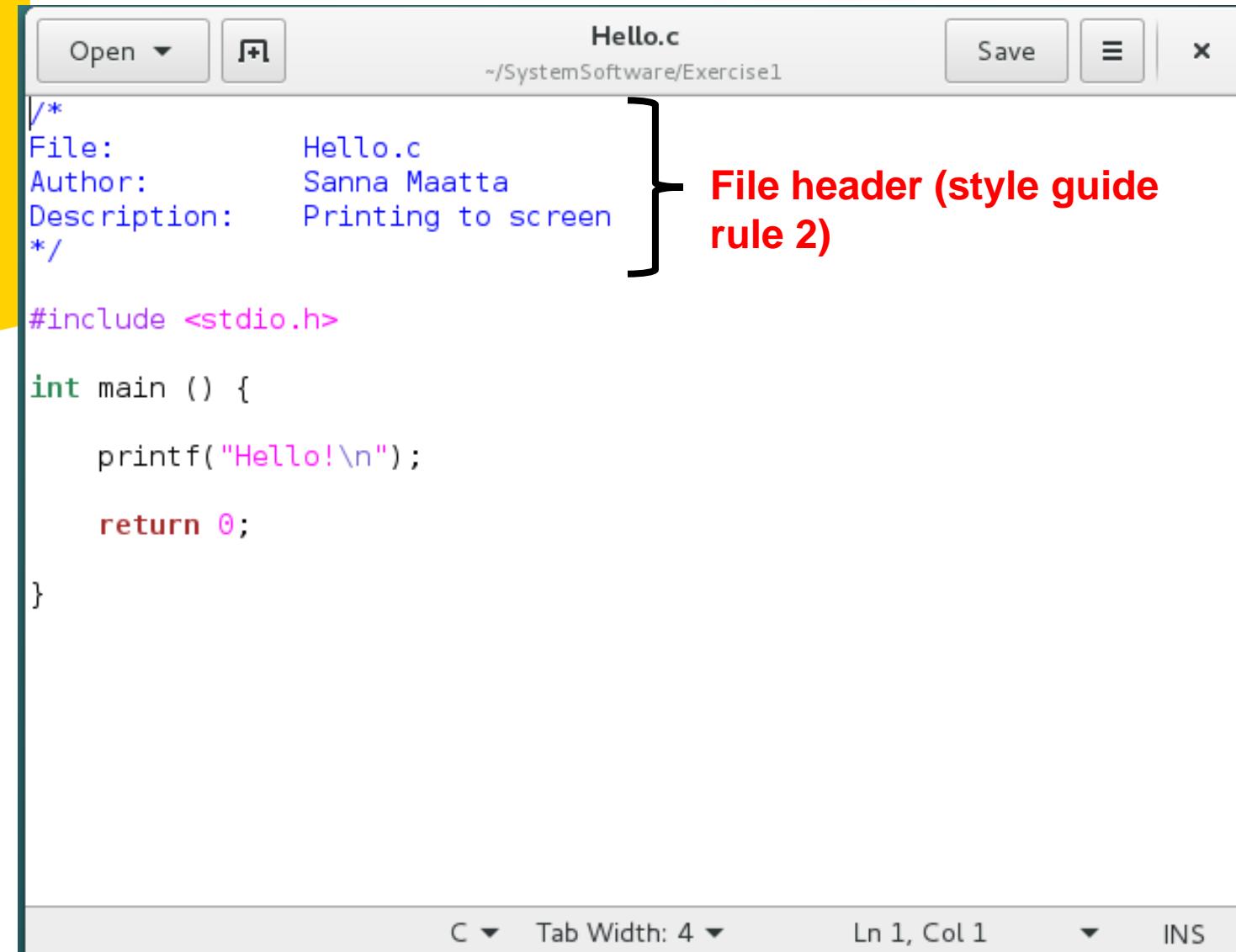
int main () {
    printf("Hello!\n");
    return 0;
}
```

C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

For comparison, same in Python.

```
1 # File: main.py
2 # Author: Sanna Maatta
3 # Printing to screen
4
5 print("Hello")
```

# First Program in C



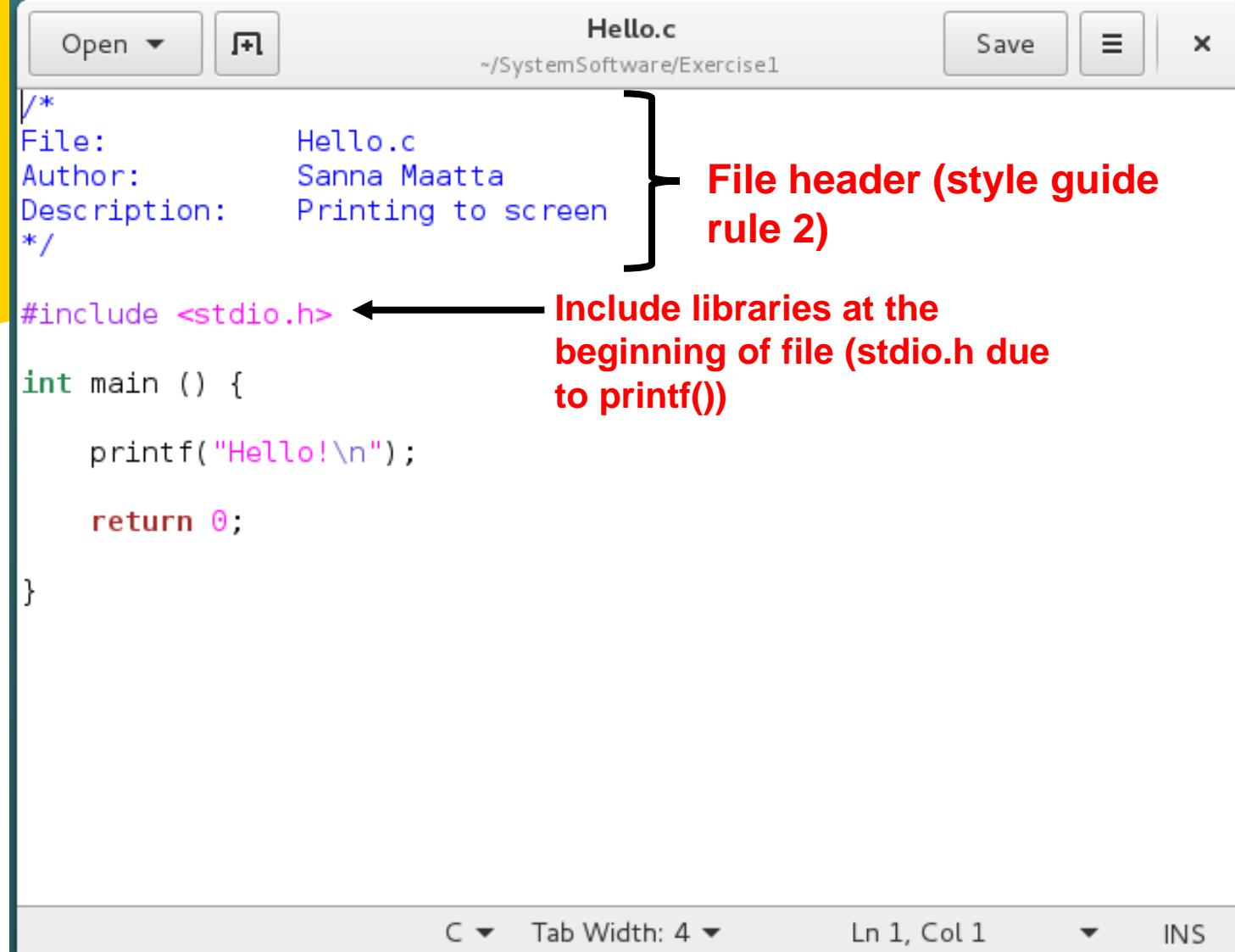
```
Open + Hello.c ~/SystemSoftware/Exercise1 Save x
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

File header (style guide rule 2)

# First Program in C



```
/*  
File:          Hello.c  
Author:        Sanna Maatta  
Description:   Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
  
    return 0;  
}
```

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf())

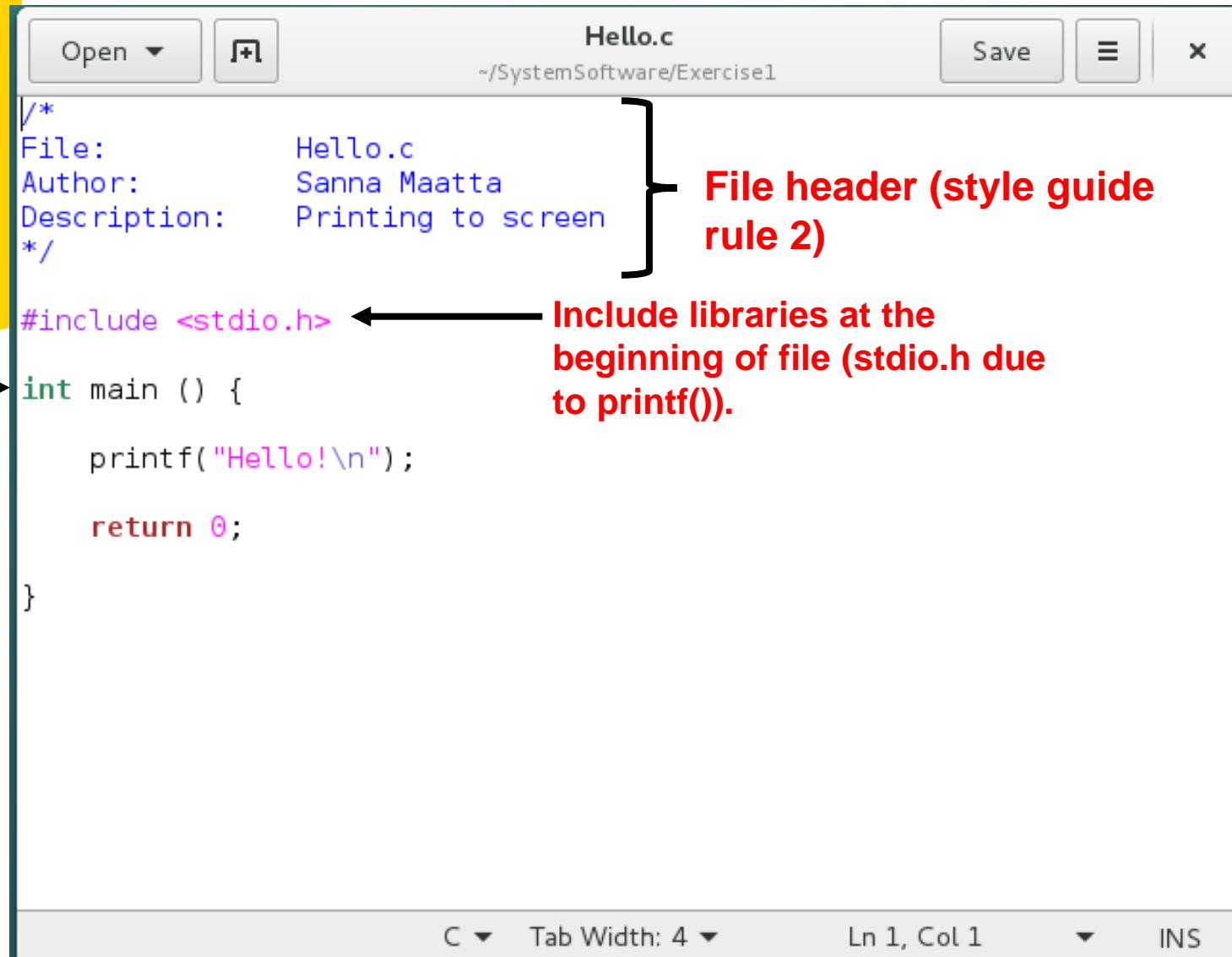
C ▾ Tab Width: 4 ▾

Ln 1, Col 1 ▾

INS

# First Program in C

Main function  
Each C program shall  
have 1 main function.



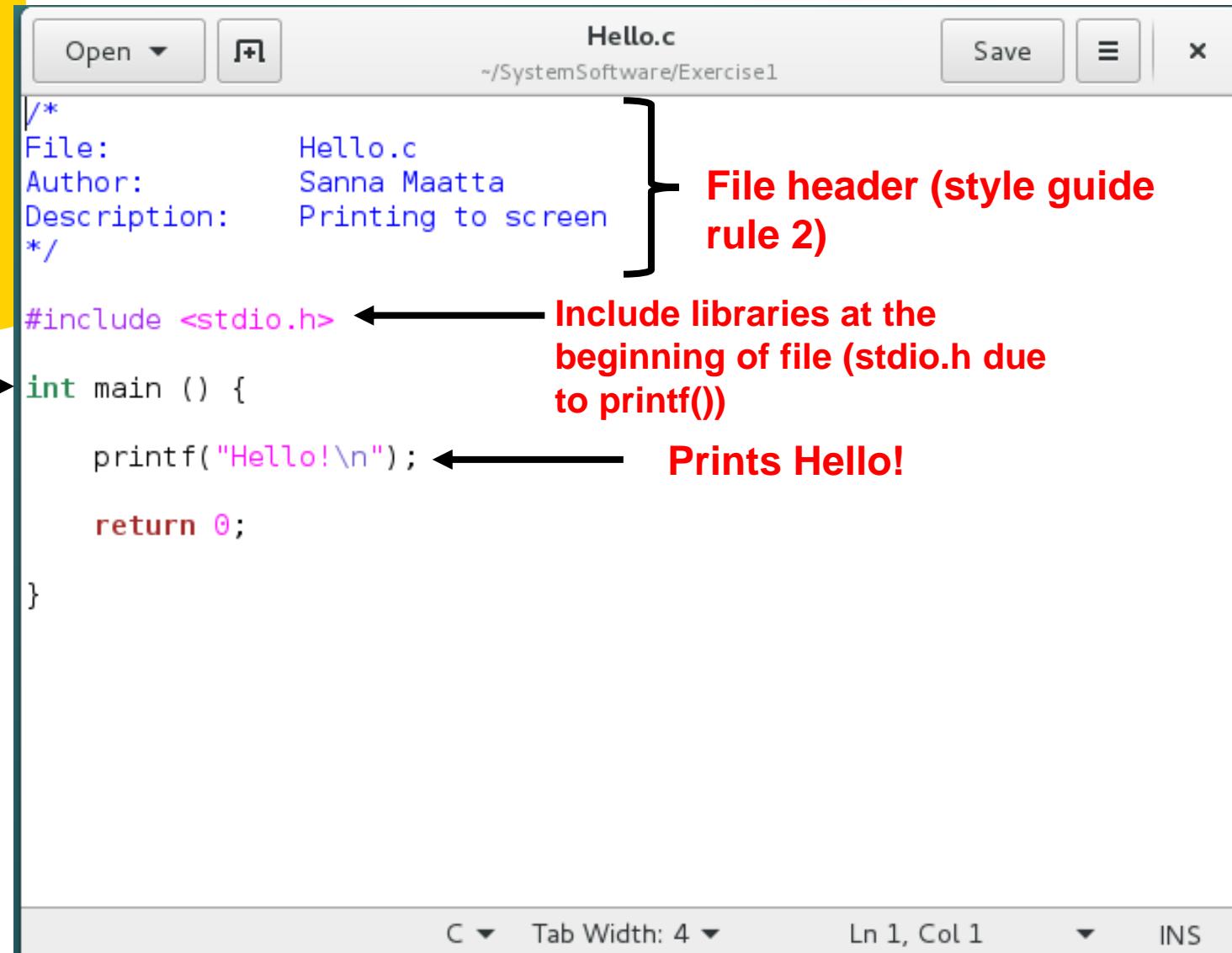
```
/*  
File:          Hello.c  
Author:        Sanna Maatta  
Description:   Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
  
    return 0;  
}
```

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf()).

# First Program in C

Main function  
Each C program shall  
have 1 main function



```
/*  
File:          Hello.c  
Author:        Sanna Maatta  
Description:   Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
  
    return 0;  
}
```

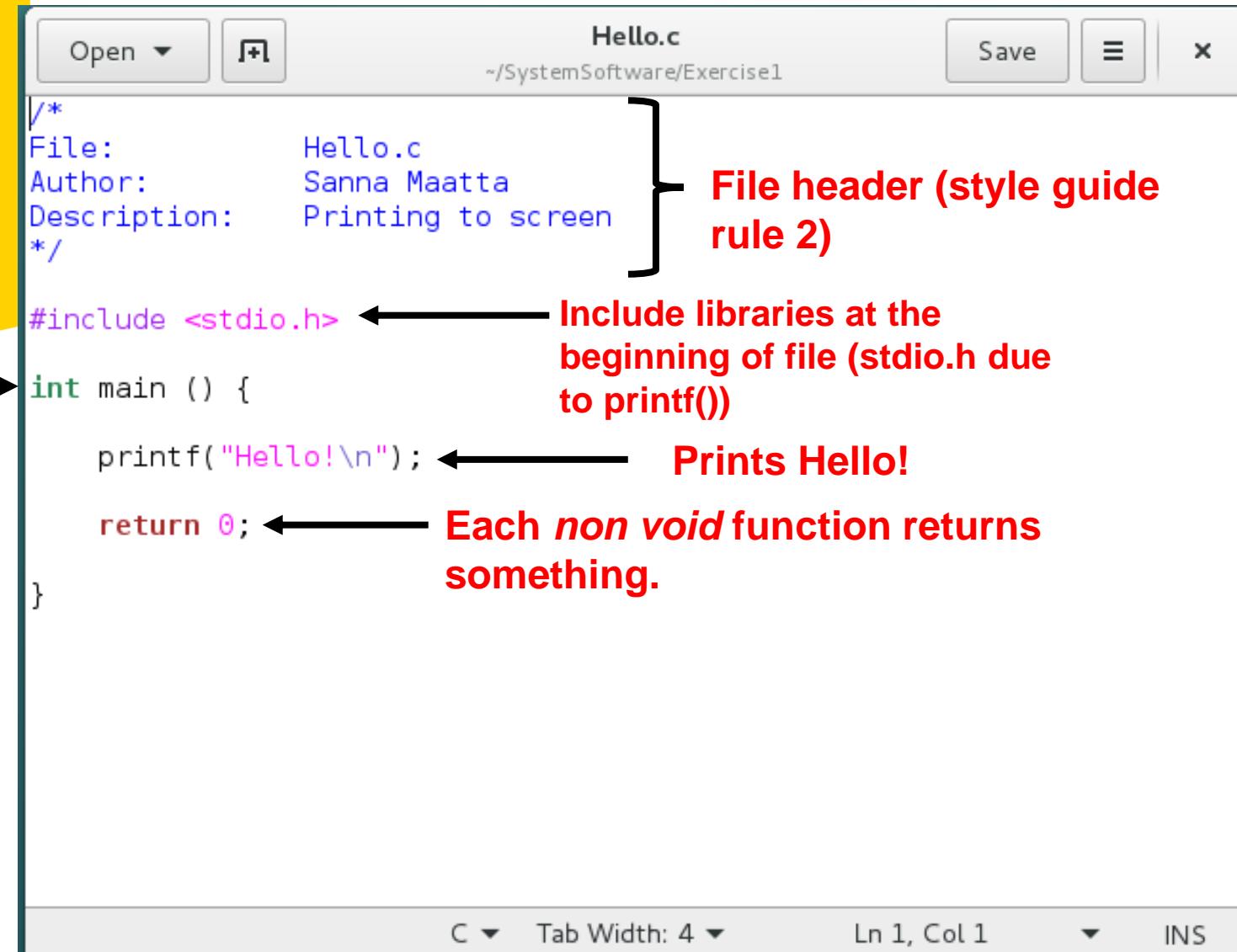
File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf())

Prints Hello!

# First Program in C

Main function  
Each C program shall  
have 1 main function



```
/*  
File:          Hello.c  
Author:        Sanna Maatta  
Description:   Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
  
    return 0;  
}
```

File header (style guide rule 2)

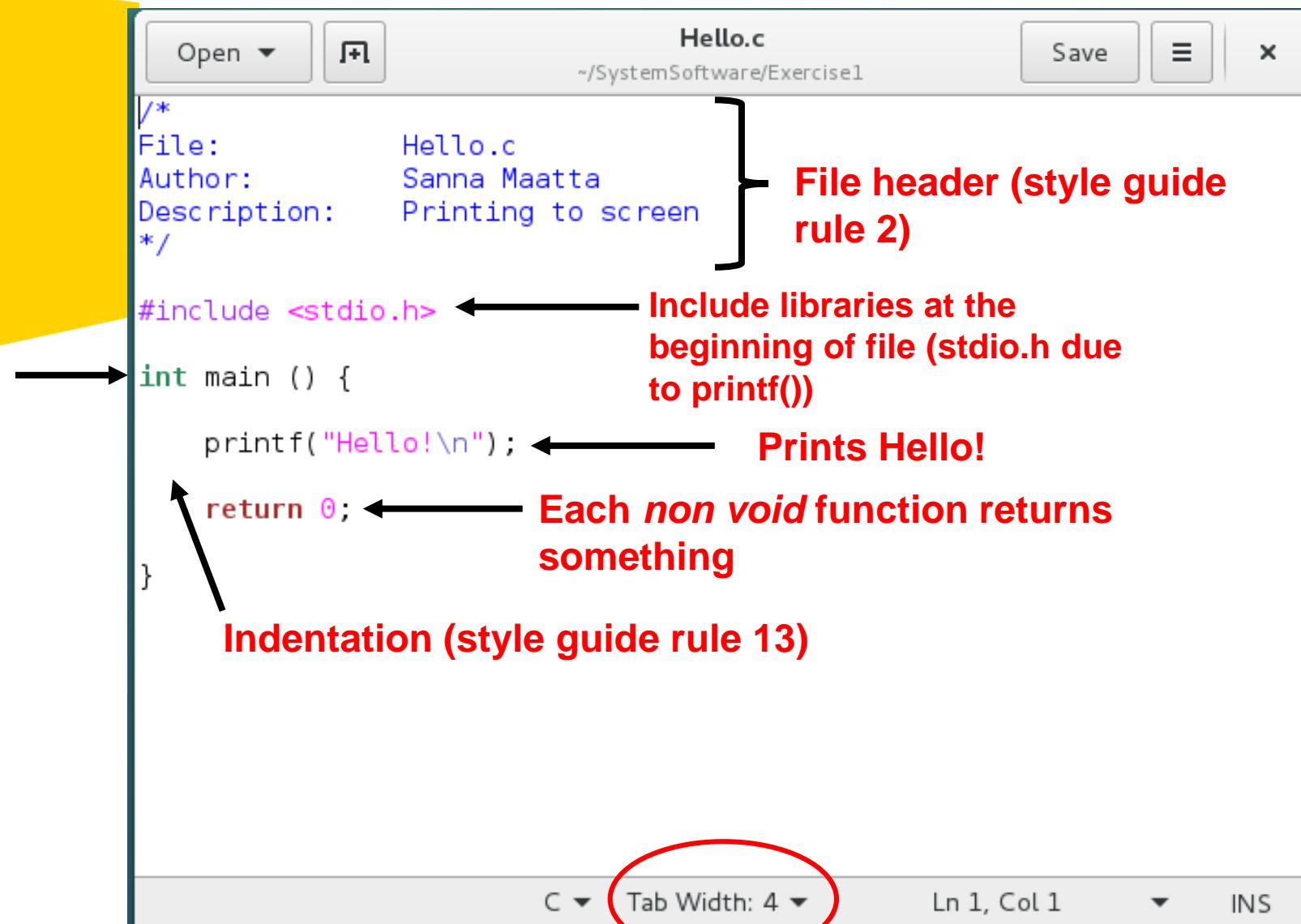
Include libraries at the beginning of file (stdio.h due to printf())

Prints Hello!

Each *non void* function returns something.

# First Program in C

Main function  
Each C program shall  
have 1 main function



```
/*  
File:          Hello.c  
Author:        Sanna Maatta  
Description:   Printing to screen  
*/  
  
#include <stdio.h> ← Include libraries at the beginning of file (stdio.h due to printf())  
  
int main () {  
    printf("Hello!\n"); ← Prints Hello!  
    return 0; ← Each non void function returns something  
}  
  
Indentation (style guide rule 13)  
  
C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS
```

# First Program in C

Main function  
Each C program shall  
have 1 main function

File naming (style guide rule 5)

```
/*  
File:          Hello.c  
Author:        Sanna Maatta  
Description:   Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
    return 0;  
}
```

File naming (style guide rule 5)

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf())

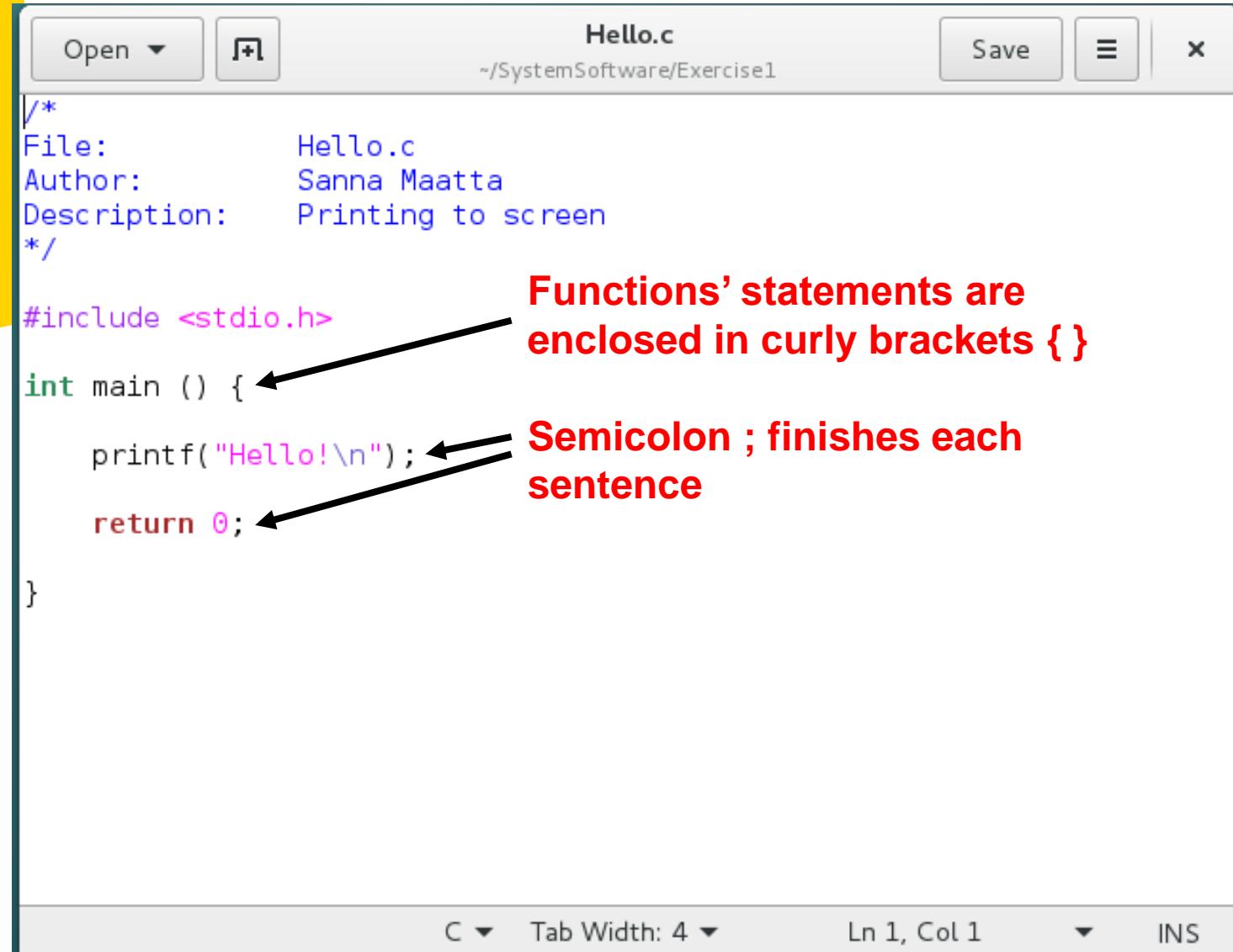
Prints Hello!

Each *non void* function returns something

Indentation (style guide rule 13)

Tab Width: 4

# First Program in C



The screenshot shows a code editor window titled "Hello.c" with the following C code:

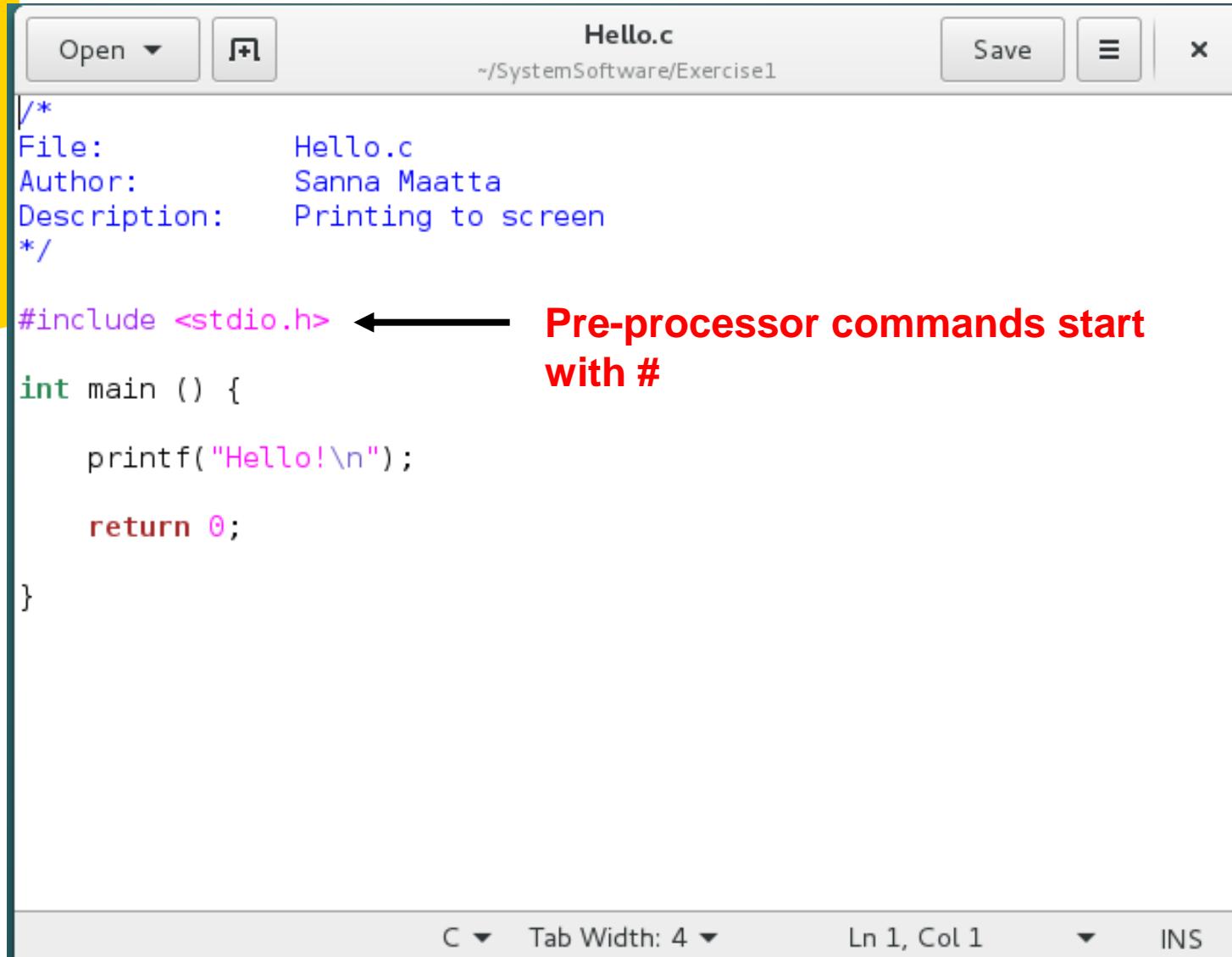
```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>
int main () {
    printf("Hello!\n");
    return 0;
}
```

Annotations in red text explain parts of the code:

- "Functions' statements are enclosed in curly brackets { }" points to the opening brace `{` after "main".
- "Semicolon ; finishes each sentence" points to the semicolon `;` after "printf".

Bottom status bar: C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

# First Program in C



```
Open ▾  Hello.c
~/SystemSoftware/Exercise1 Save ⌂ x

/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/

#include <stdio.h> ← Pre-processor commands start with #

int main () {
    printf("Hello!\n");
    return 0;
}

C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS
```

# How to Compile and Run C Code on Linux

# Compiling and Running on Linux

The image shows a Linux desktop environment with a terminal window and a code editor window.

**Code Editor (gedit):**

```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

**Terminal:**

```
sanna@debian: ~/SystemSoftware/Exercise1
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

# Compiling and Running on Linux

The screenshot shows a Linux desktop environment. In the top right corner, there is a window titled "Hello.c" which is an open file in the Gedit text editor. The code in the file is:

```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>
int main () {
    printf("Hello!\n");
    return 0;
}
```

In the bottom right corner, there is a terminal window with the command line interface. The terminal shows the user's session:

```
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

Two red annotations are present: one highlights the command "gedit Hello.c &" in the terminal, and another annotation points from this highlight to the explanatory text below.

**Open/create file “Hello.c” in Gedit editor**

**& (ampersand) is not mandatory but allows you to use terminal while file is open, so use it!**

# Compiling and Running on Linux

The image shows a Linux desktop environment with a terminal window and a code editor window.

**Code Editor (gedit):**

```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

**Terminal:**

```
sanna@debian: ~/SystemSoftware/Exercise1
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

A red box highlights the command `gcc -Wall -Wextra -std=c18 Hello.c -o Hello` in the terminal window.

**Text on the right:**

**gcc is the compiler (Gnu Compiler Collection)**  
**-Wall -Wextra -std=c18 are compiler options**

# Compiling and Running on Linux

The image shows a Linux desktop environment with a terminal window and a code editor window.

**Code Editor (gedit):**

```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

**Terminal:**

```
sanna@debian: ~/SystemSoftware/Exercise1
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

A red box highlights the command `gcc -Wall -Wextra -std=c18 Hello.c -o Hello` in the terminal, and a black arrow points from this box to the explanatory text below.

**-o makes an executable “Hello” from Hello.c**

# Compiling and Running on Linux

The image shows a Linux desktop environment with a terminal window and a code editor window.

**Code Editor (gedit):**

```
/*
File:          Hello.c
Author:        Sanna Maatta
Description:   Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

**Terminal:**

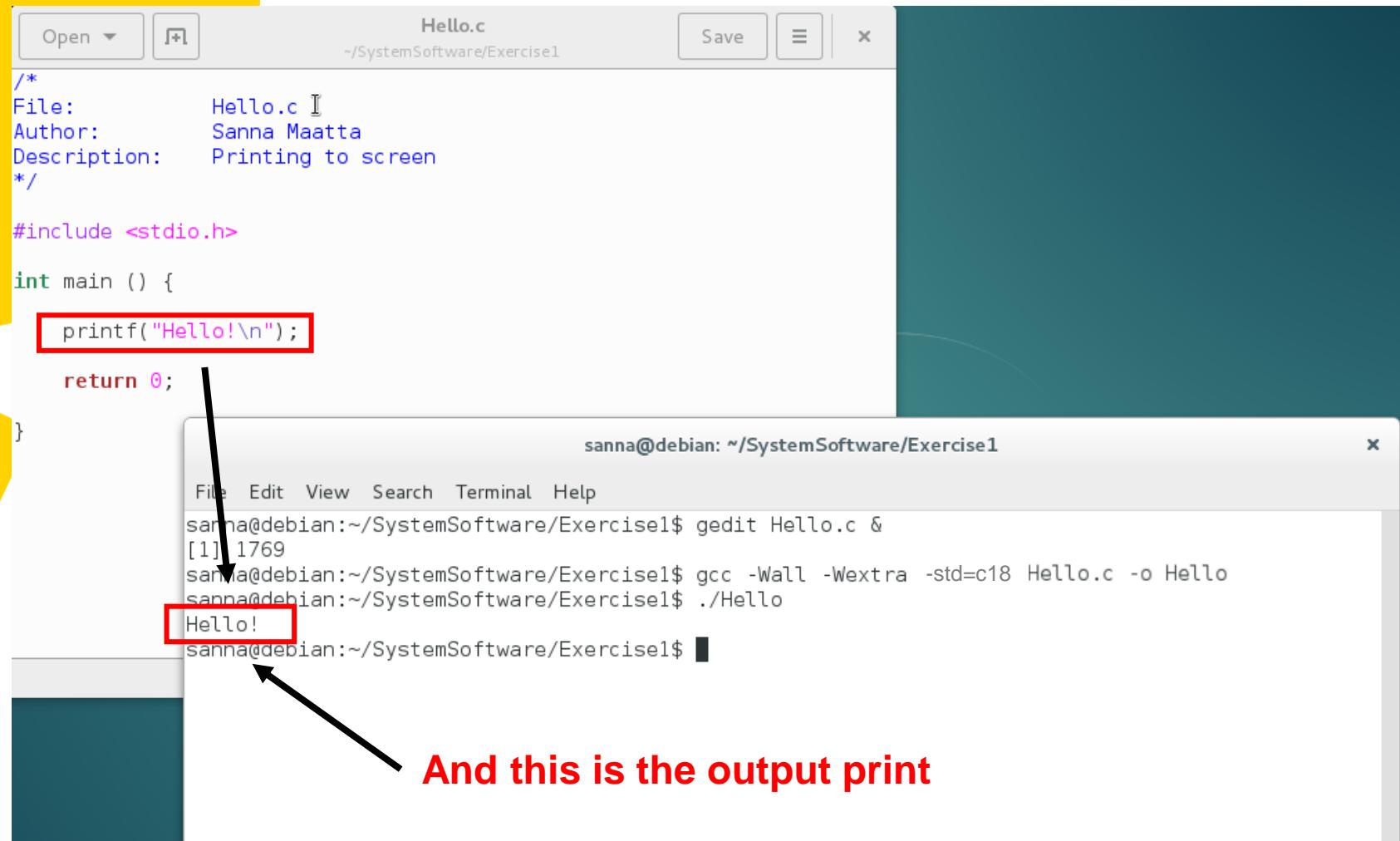
```
sanna@debian: ~/SystemSoftware/Exercise1
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

A red box highlights the command `./Hello` in the terminal, and a black arrow points from this box to the text "Running the executable program" below.

**Text Labels:**

- Running the executable program**
- Notice, there is no file extension .c  
any more in the executable's name**

# Compiling and Running on Linux



The image shows a terminal window titled "sanna@debian: ~/SystemSoftware/Exercise1". The window displays the following command-line session:

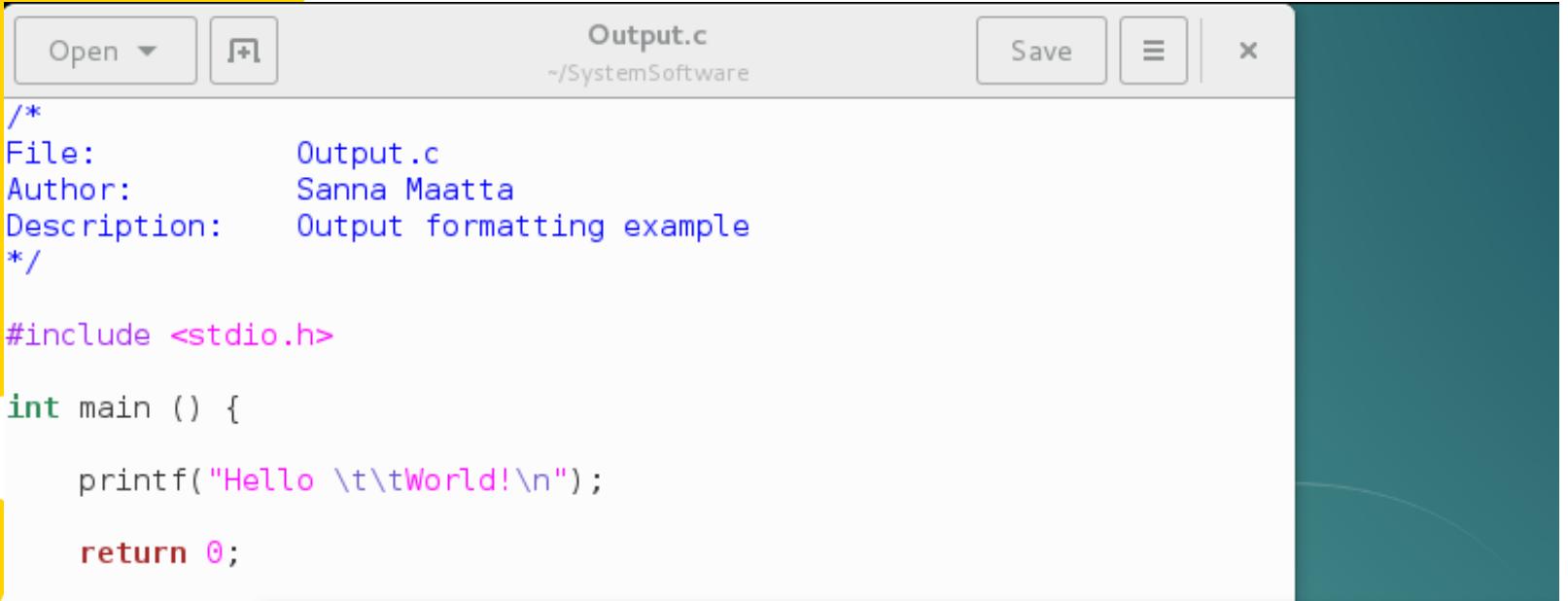
```
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

A red box highlights the word "Hello!" in the terminal output. A black arrow points from this highlighted text to the following red text at the bottom:

**And this is the output print**

# Output Formatting in C

# Output Formatting Example 1

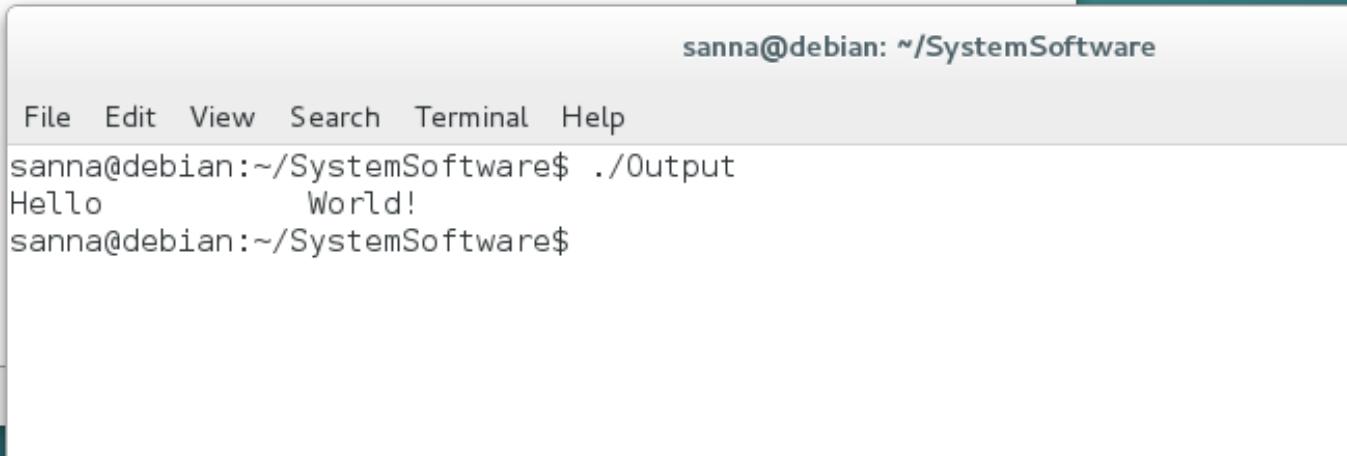


```
Open + Output.c ~/SystemSoftware Save x
/*
File:          Output.c
Author:        Sanna Maatta
Description:   Output formatting example
*/
#include <stdio.h>

int main () {

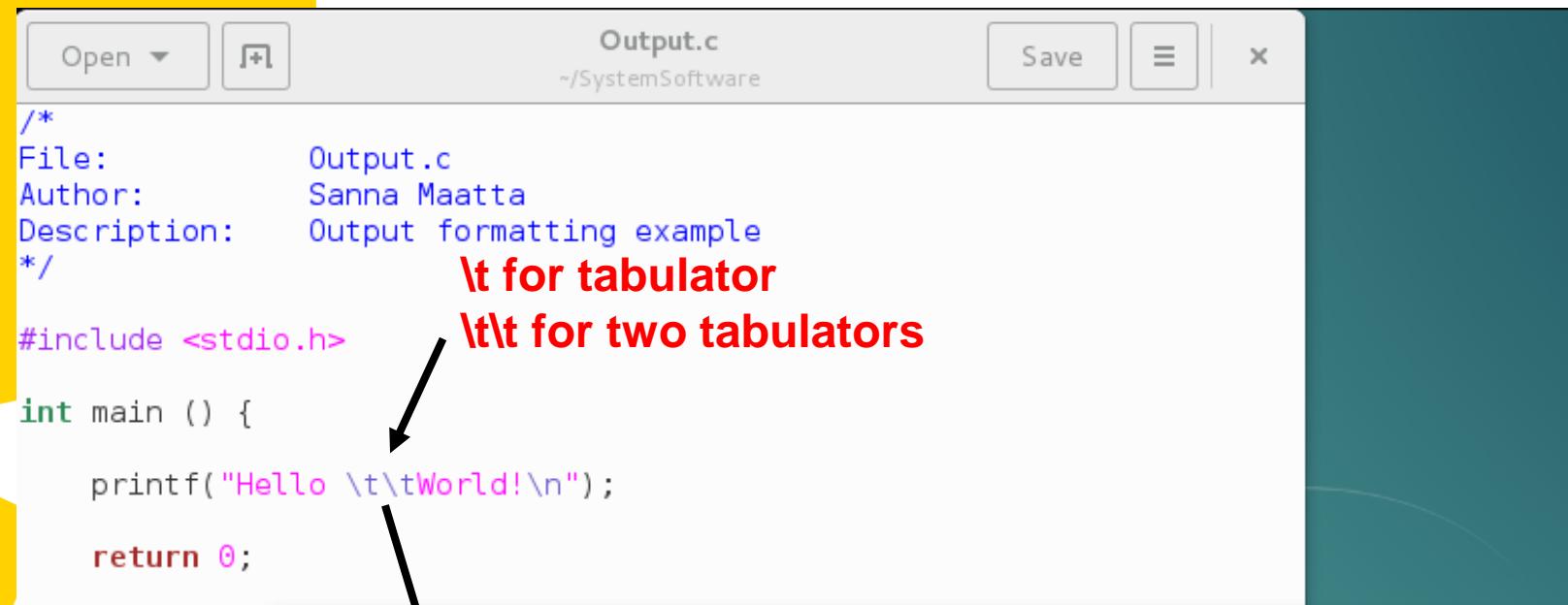
    printf("Hello \t\tWorld!\n");

    return 0;
}
```



```
sanna@debian: ~/SystemSoftware
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello      World!
sanna@debian:~/SystemSoftware$
```

# Output Formatting Example 1

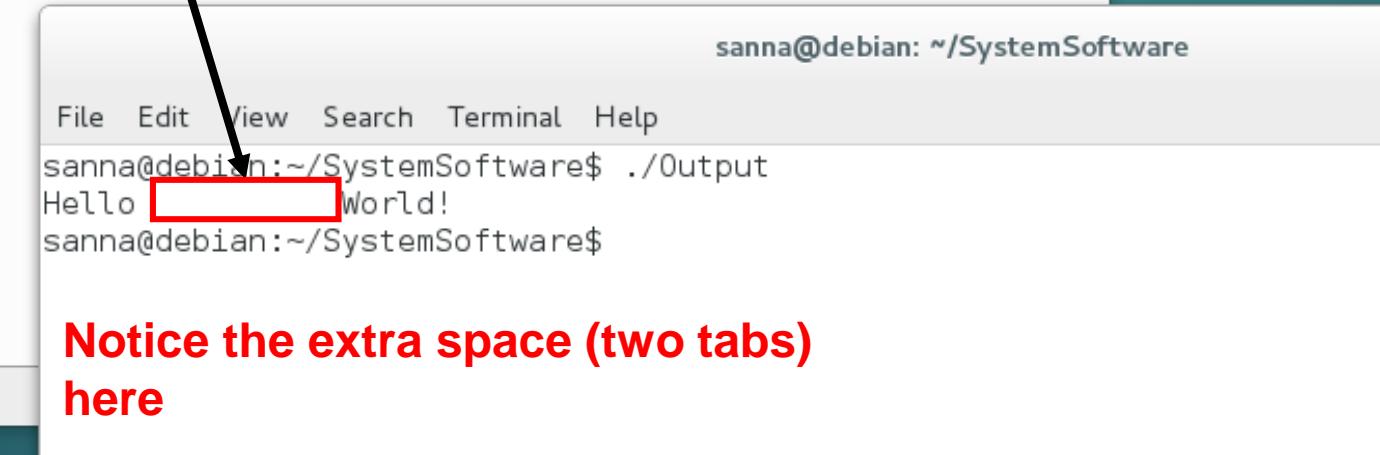


```
Output.c
~/SystemSoftware
Open + Save x
/*
File:          Output.c
Author:        Sanna Maatta
Description:   Output formatting example
*/
#include <stdio.h>
int main () {
    printf("Hello \t\tWorld!\n");
    return 0;
}
```

The code editor window shows a C program named `Output.c`. The code includes a multi-line comment at the top and a `printf` statement in the `main` function. The terminal window below shows the output of running the program, which is "Hello World!". The terminal window has a title bar "sanna@debian: ~/SystemSoftware".

**\t for tabulator**

**\t\t for two tabulators**



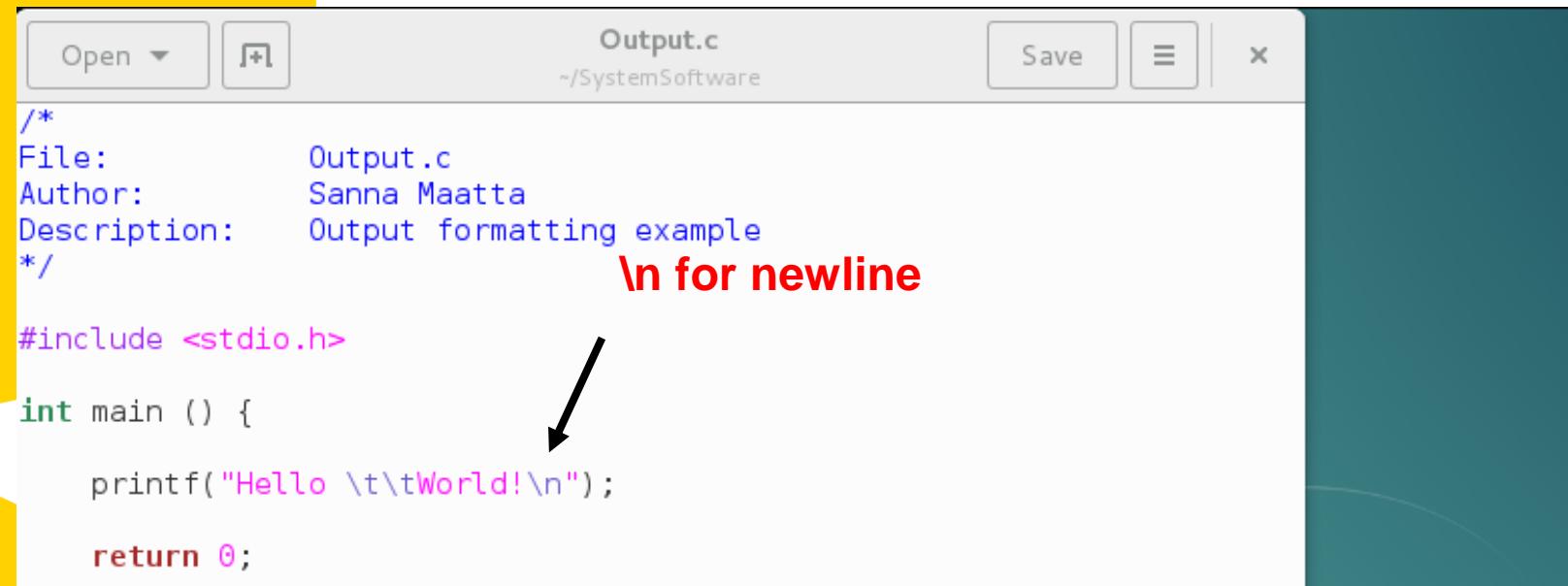
```
sanna@debian: ~/SystemSoftware
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello [redacted]World!
sanna@debian:~/SystemSoftware$
```

The terminal window shows the command `./Output` being run and the resulting output "Hello [redacted]World!". The word "Hello" and "World!" are aligned horizontally, demonstrating the use of tabs for output formatting.

**Notice the extra space (two tabs)  
here**

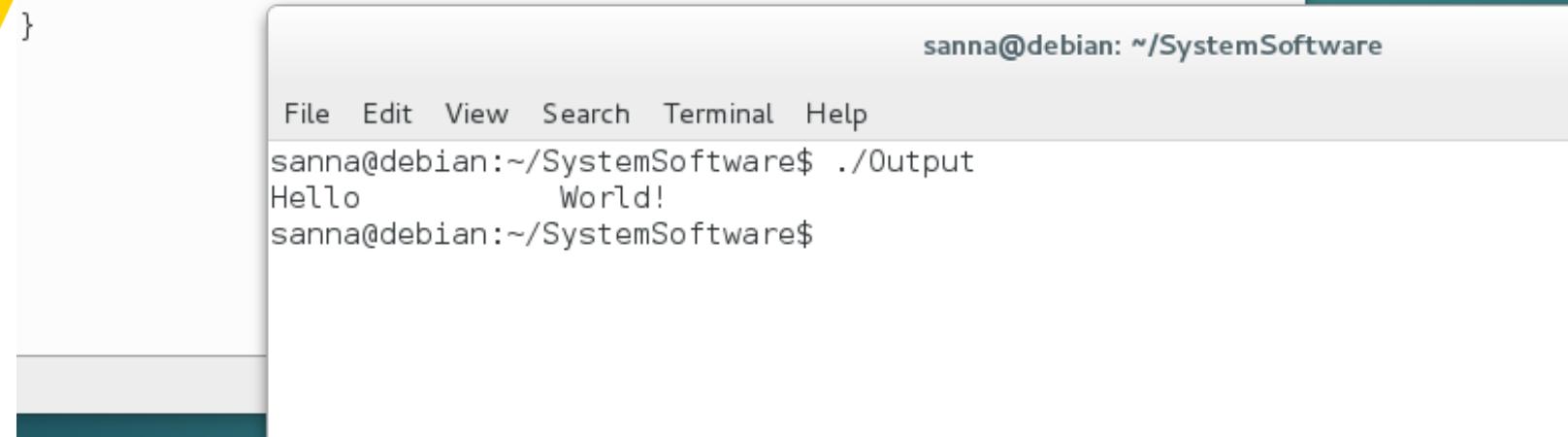
# Output Formatting Example 1

- \n (newline)
- \t (tab)
- \v (vertical tab)
- \f (new page)
- \b (backspace)
- \r (carriage return)



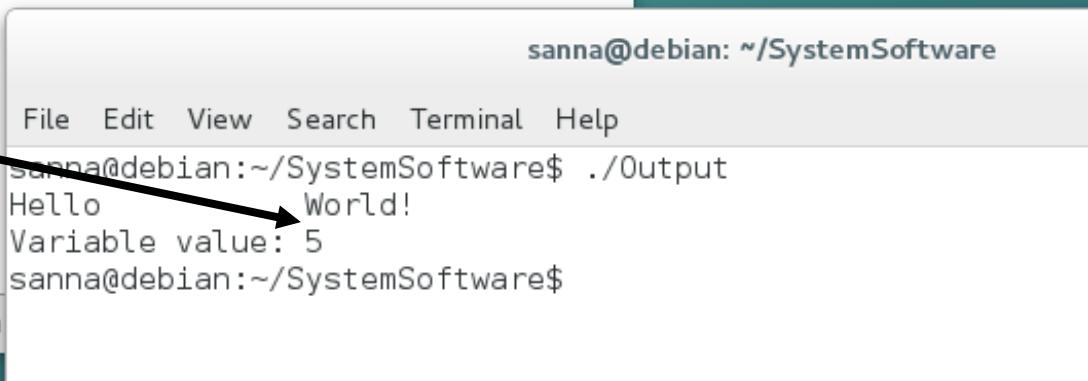
```
Output.c
~/SystemSoftware
Open + Save X
/*
File:      Output.c
Author:    Sanna Maatta
Description: Output formatting example
*/
#include <stdio.h>

int main () {
    printf("Hello \t\tWorld!\n");
    return 0;
}
```



```
sanna@debian: ~/SystemSoftware
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello
World!
sanna@debian:~/SystemSoftware$
```

# Output Formatting Example 2



The terminal window shows the execution of the program and its output:

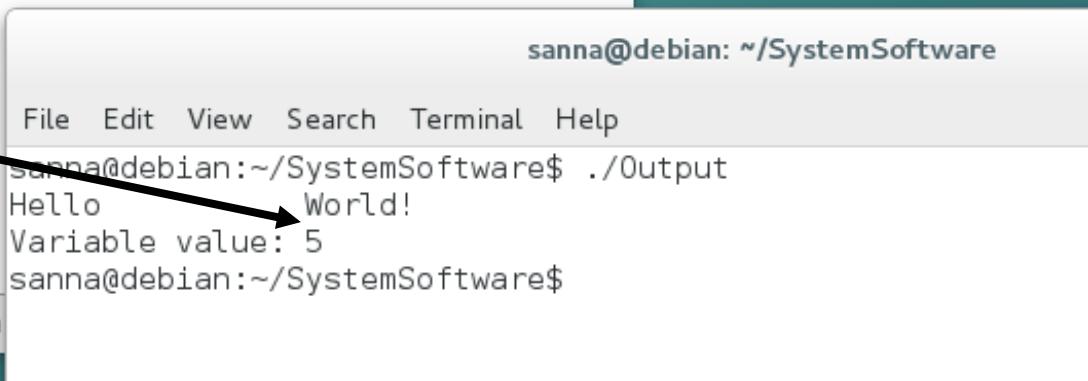
```
sanna@debian: ~/SystemSoftware$ ./Output
Hello      World!
Variable value: 5
sanna@debian: ~/SystemSoftware$
```

A red callout box with the text "Notice the syntax when printing out variable values." points to the printf statement that outputs the variable value.

```
/*
File:          Output.c
Author:        Sanna Maatta
Description:   Output formatting example
*/
#include <stdio.h>

int main () {
    int testVariable = 5;
    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d\n", testVariable);
    return 0;
}
```

# Output Formatting Example 2



The terminal window shows the execution of the program and its output:

```
sanna@debian: ~/SystemSoftware$ ./Output
Hello      World!
Variable value: 5
sanna@debian: ~/SystemSoftware$
```

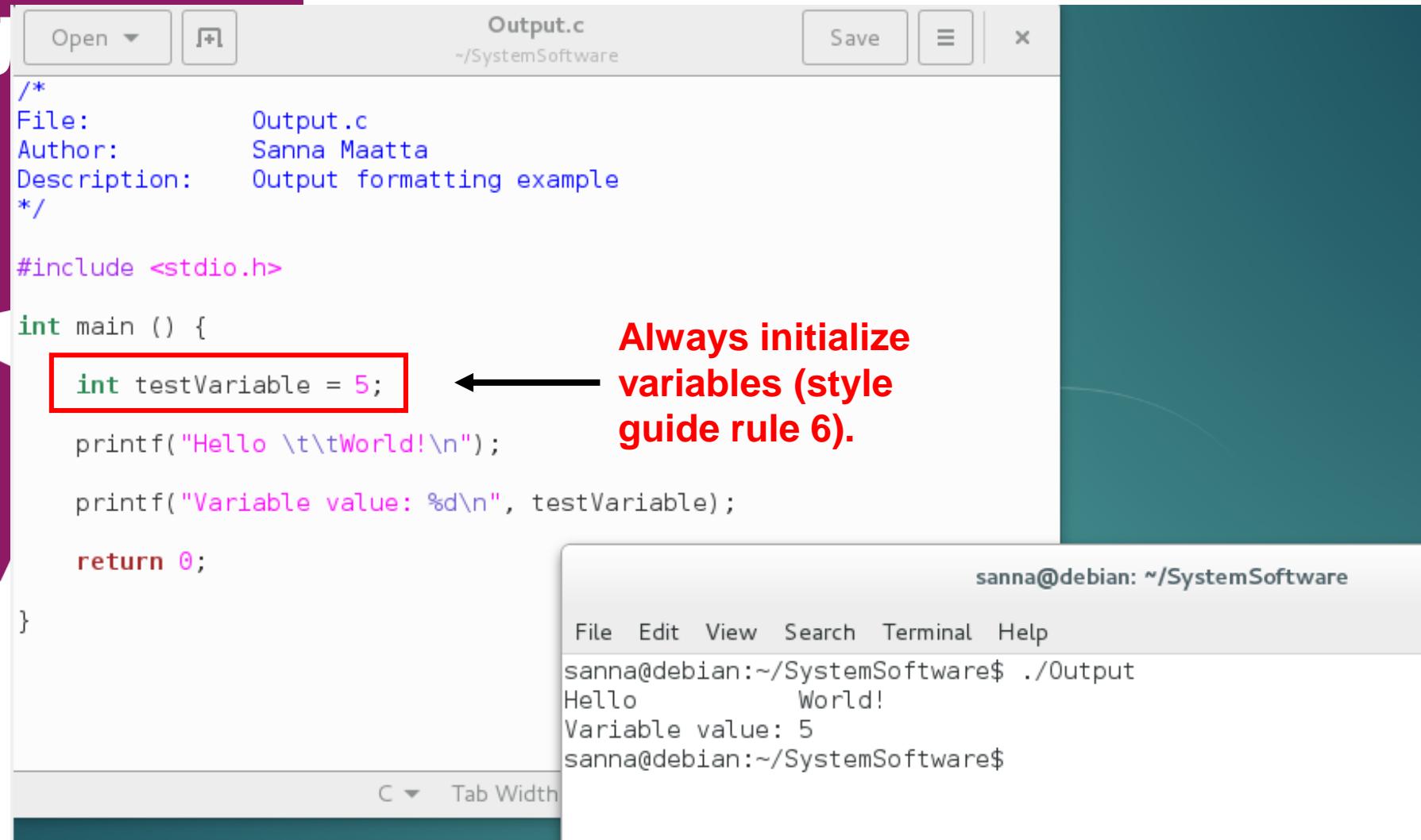
Annotations in red text and arrows point to the printf statements in the code and the corresponding output lines in the terminal.

**%d: integer**  
**%c: character**  
**%f: float and double**  
**%x: hexadecimal**

Notice the syntax when printing out variable values.

```
/*
File:          Output.c
Author:        Sanna Maatta
Description:   Output formatting example
*/
#include <stdio.h>
int main () {
    int testVariable = 5;
    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d\n", testVariable);
    return 0;
}
```

# Output Formatting Example 2



The screenshot shows a code editor window titled "Output.c" with the following content:

```
/*
File:          Output.c
Author:        Sanna Maatta
Description:   Output formatting example
*/
#include <stdio.h>

int main () {
    int testVariable = 5;
    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d\n", testVariable);
    return 0;
}
```

A red box highlights the line `int testVariable = 5;`. A black arrow points from the text "Always initialize variables (style guide rule 6)." to this line.

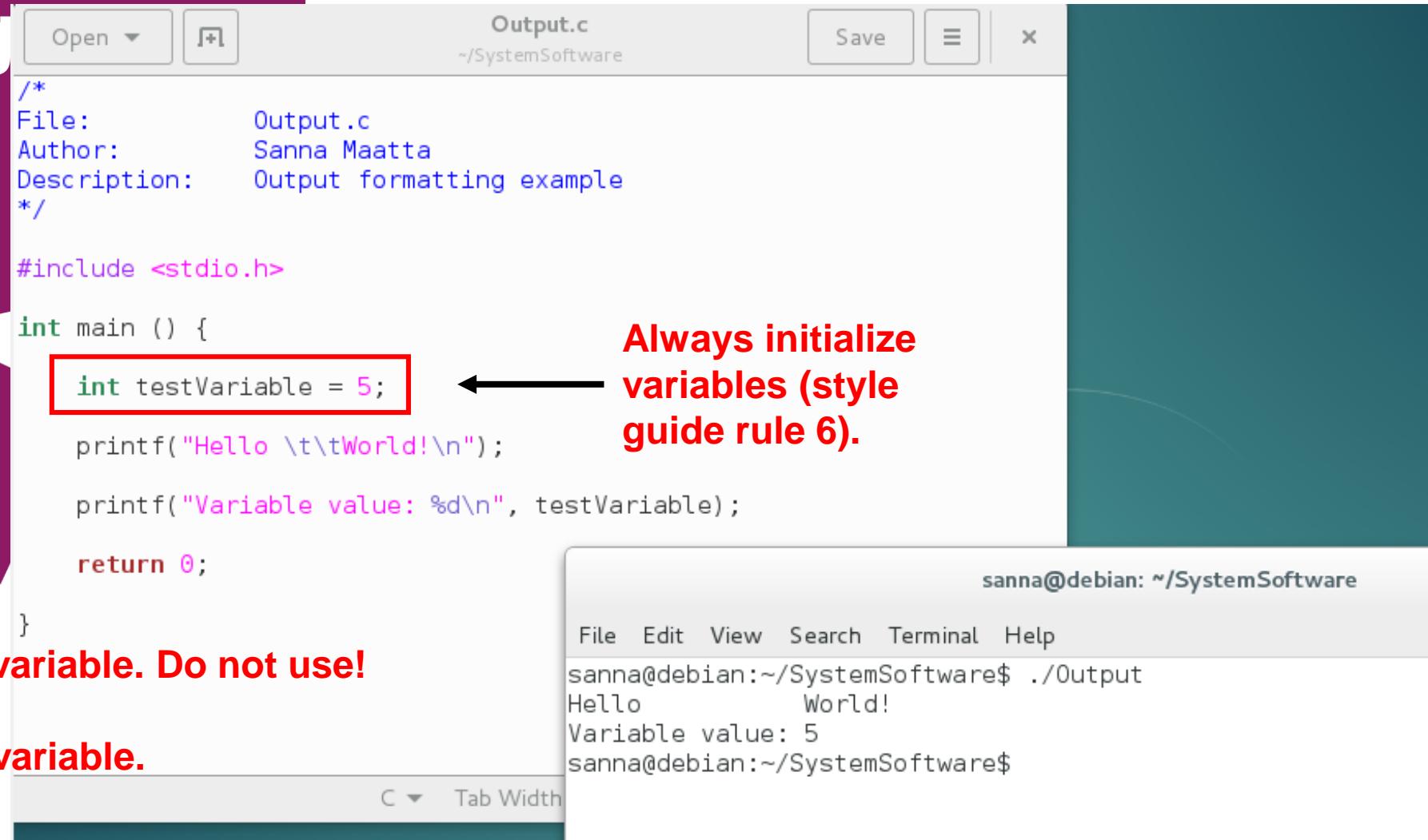
Below the code editor is a terminal window showing the execution of the program:

```
sanna@debian: ~/SystemSoftware$ ./Output
Hello           World!
Variable value: 5
sanna@debian: ~/SystemSoftware$
```

# Output Formatting Example 2

**int number; // Un-initialized variable. Do not use!**

**int number = 0; // Initialized variable.**

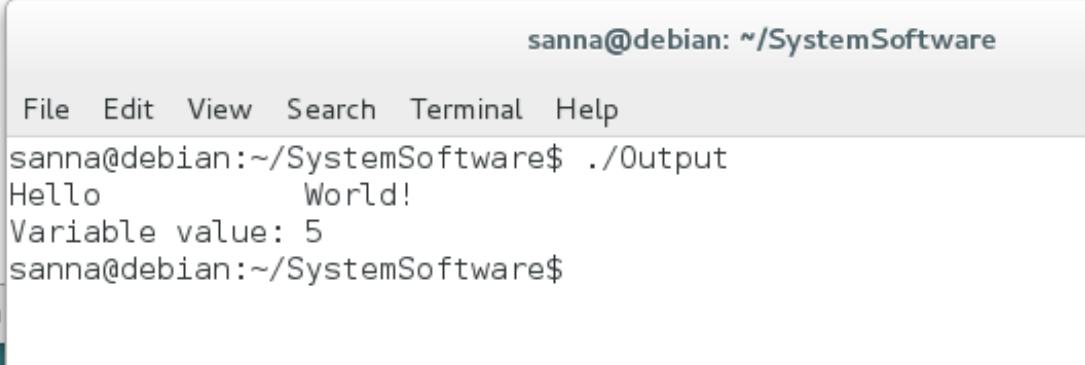


```
/*
File:          Output.c
Author:        Sanna Maatta
Description:   Output formatting example
*/

#include <stdio.h>

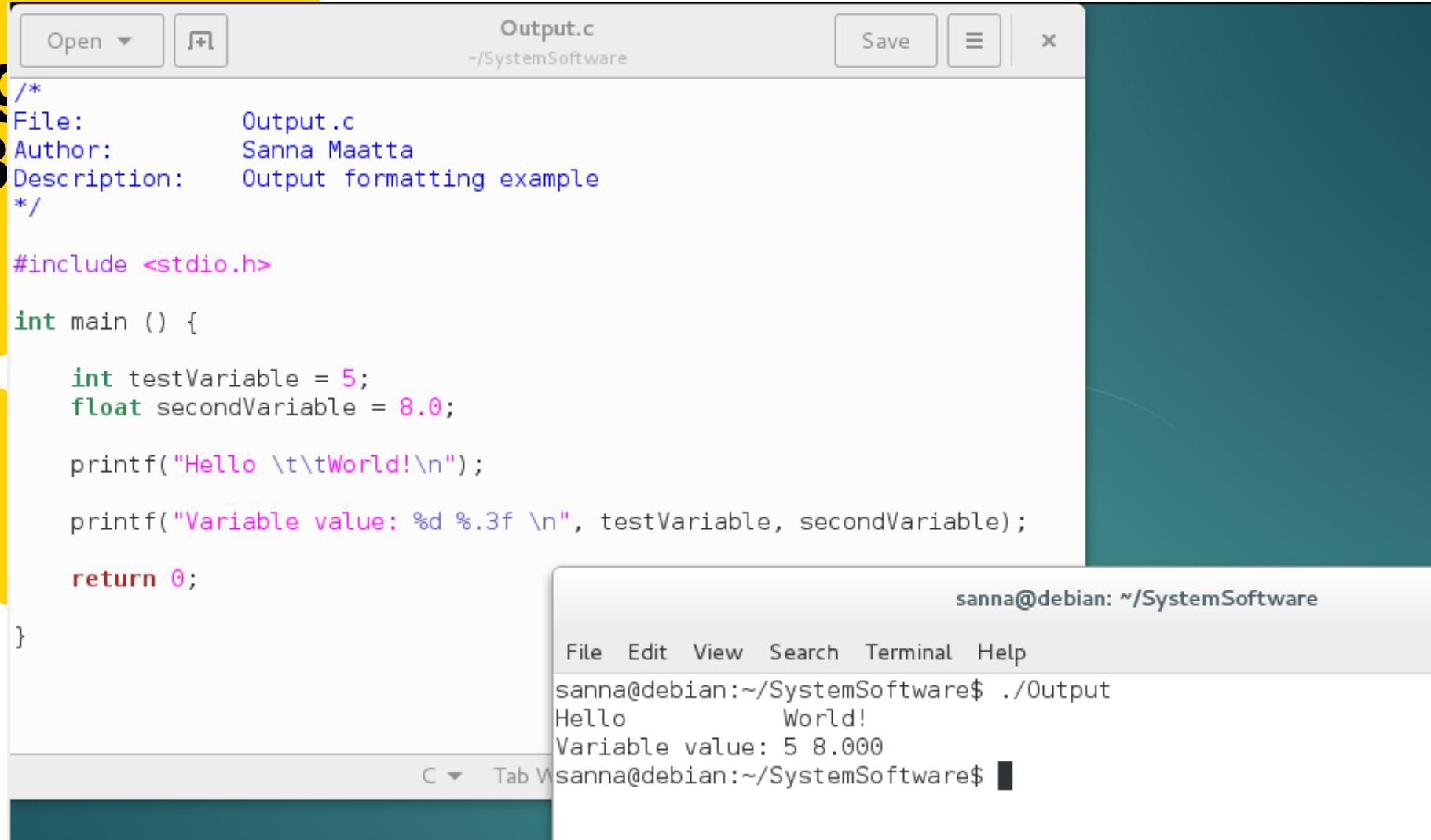
int main () {
    int testVariable = 5;
    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d\n", testVariable);
    return 0;
}
```

Always initialize variables (style guide rule 6).



```
sanna@debian: ~/SystemSoftware$ ./Output
Hello           World!
Variable value: 5
sanna@debian: ~/SystemSoftware$
```

# Output Formatting Example 3



The image shows a screenshot of a terminal window and a code editor. The terminal window at the bottom right displays the output of a C program named 'Output.c'. The code in the editor shows how to use printf() with different format specifiers to print variables. The terminal output shows the program's execution and its results.

```
Output.c
~/SystemSoftware
Save | x

/*
File:      Output.c
Author:    Sanna Maatta
Description: Output formatting example
*/

#include <stdio.h>

int main () {

    int testVariable = 5;
    float secondVariable = 8.0;

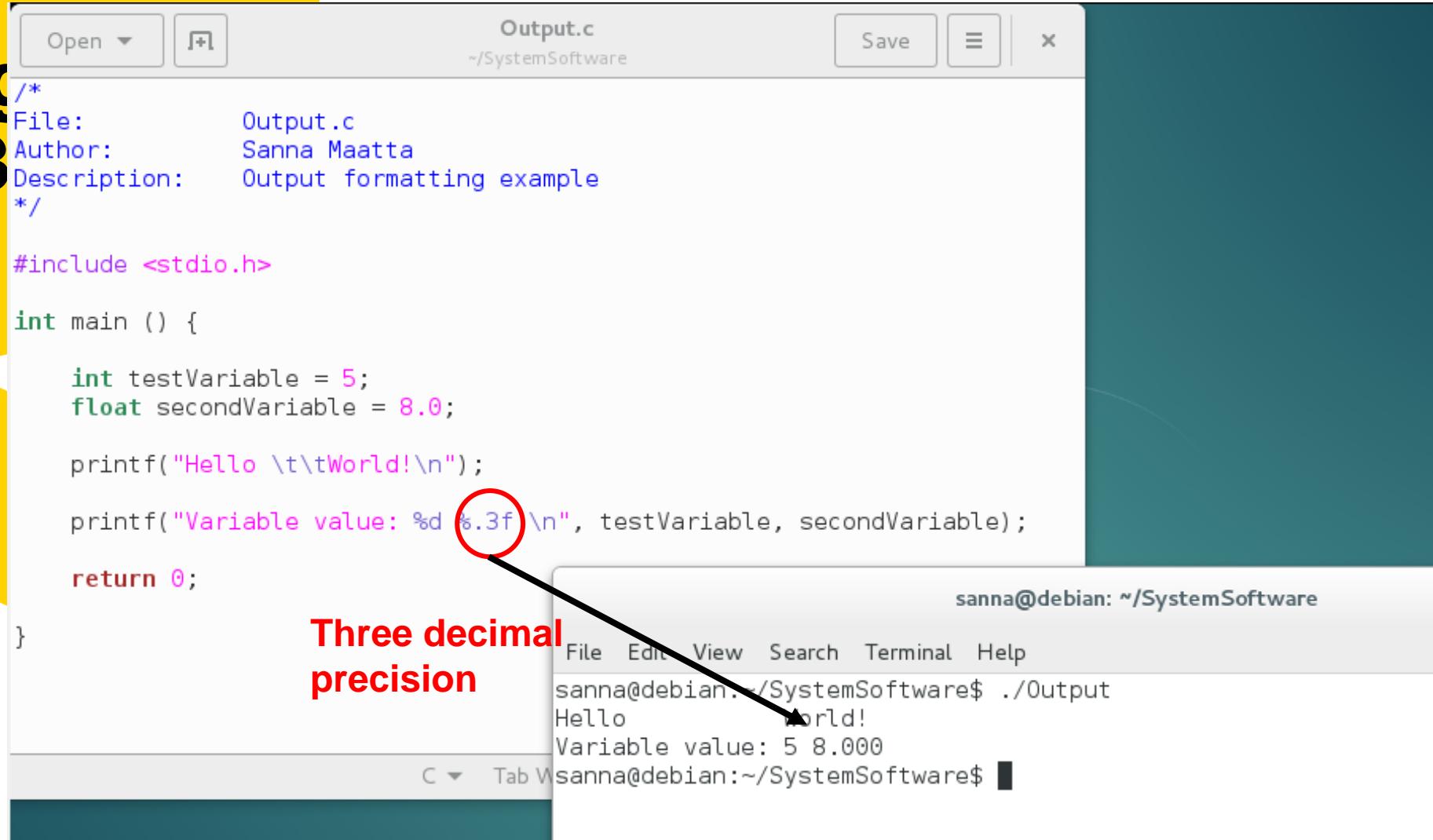
    printf("Hello \t\tWorld!\n");

    printf("Variable value: %d %.3f \n", testVariable, secondVariable);

    return 0;
}

sanna@debian: ~/SystemSoftware$ ./Output
Hello          World!
Variable value: 5 8.000
sanna@debian: ~/SystemSoftware$
```

# Output Formatting Example 3



```
Output.c
~/SystemSoftware
Open Save x
File: Output.c
Author: Sanna Maatta
Description: Output formatting example
*/
#include <stdio.h>

int main () {

    int testVariable = 5;
    float secondVariable = 8.0;

    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d %.3f\n", testVariable, secondVariable);

    return 0;
}

sanna@debian: ~/SystemSoftware
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello          World!
Variable value: 5 8.000
sanna@debian:~/SystemSoftware$
```

Three decimal precision

Question  
s?

