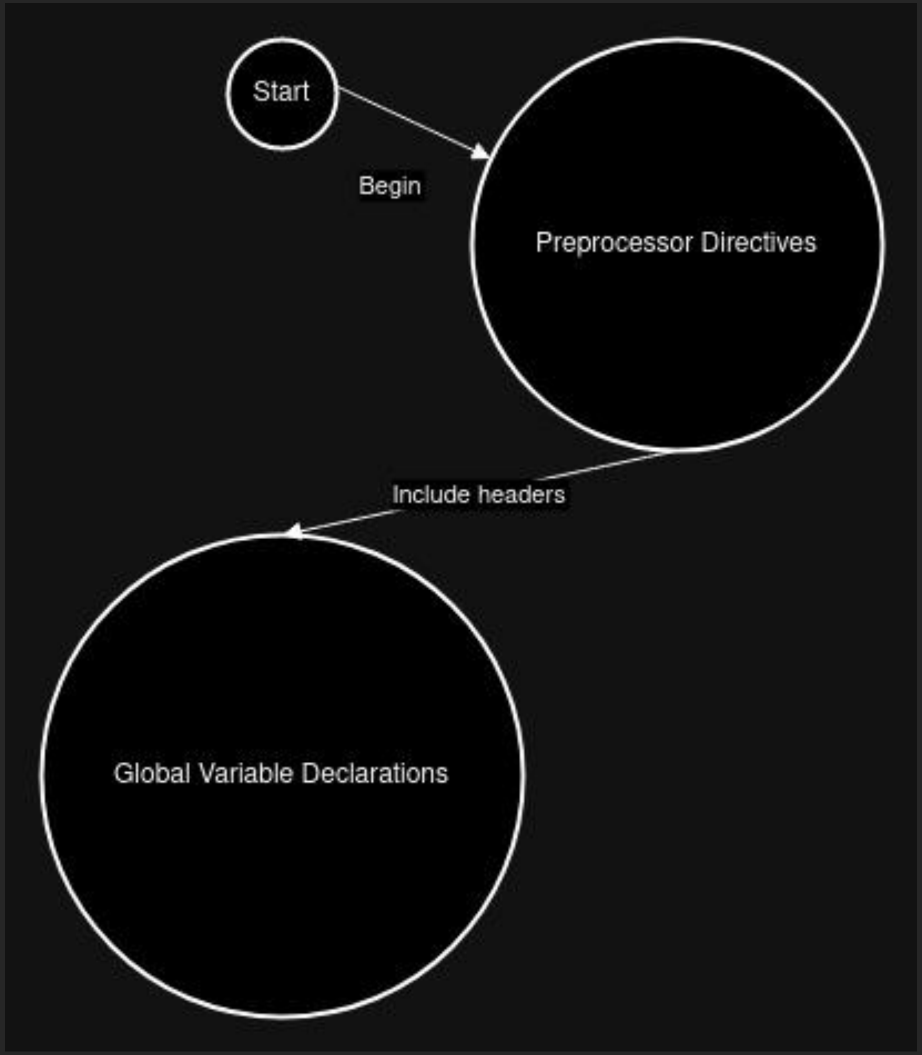
C Programming : A Deep Dive

UNLOCKING THE BUILDING BLOCKS OF COMPUTING

Kaustav

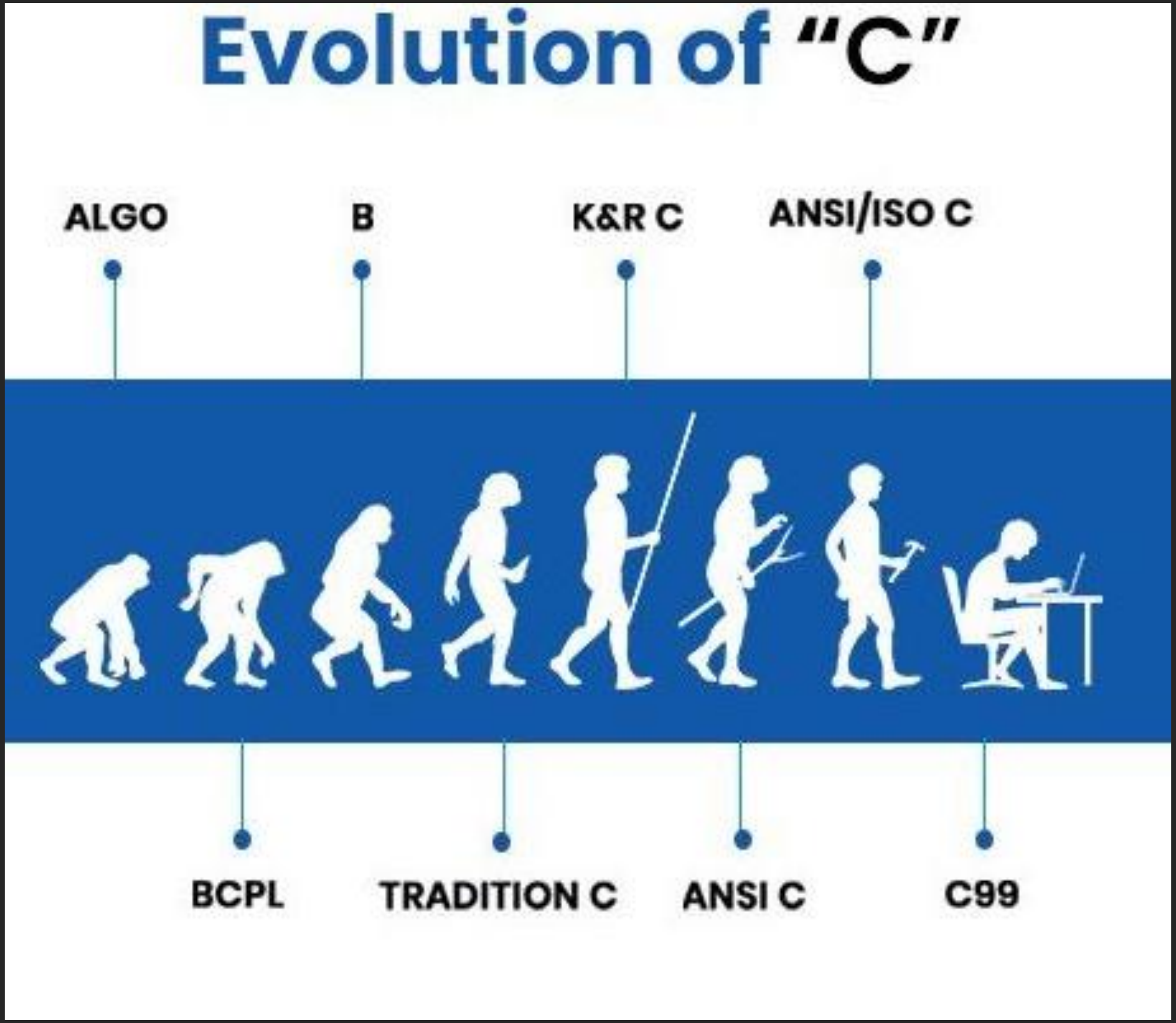
WHAT IS C?

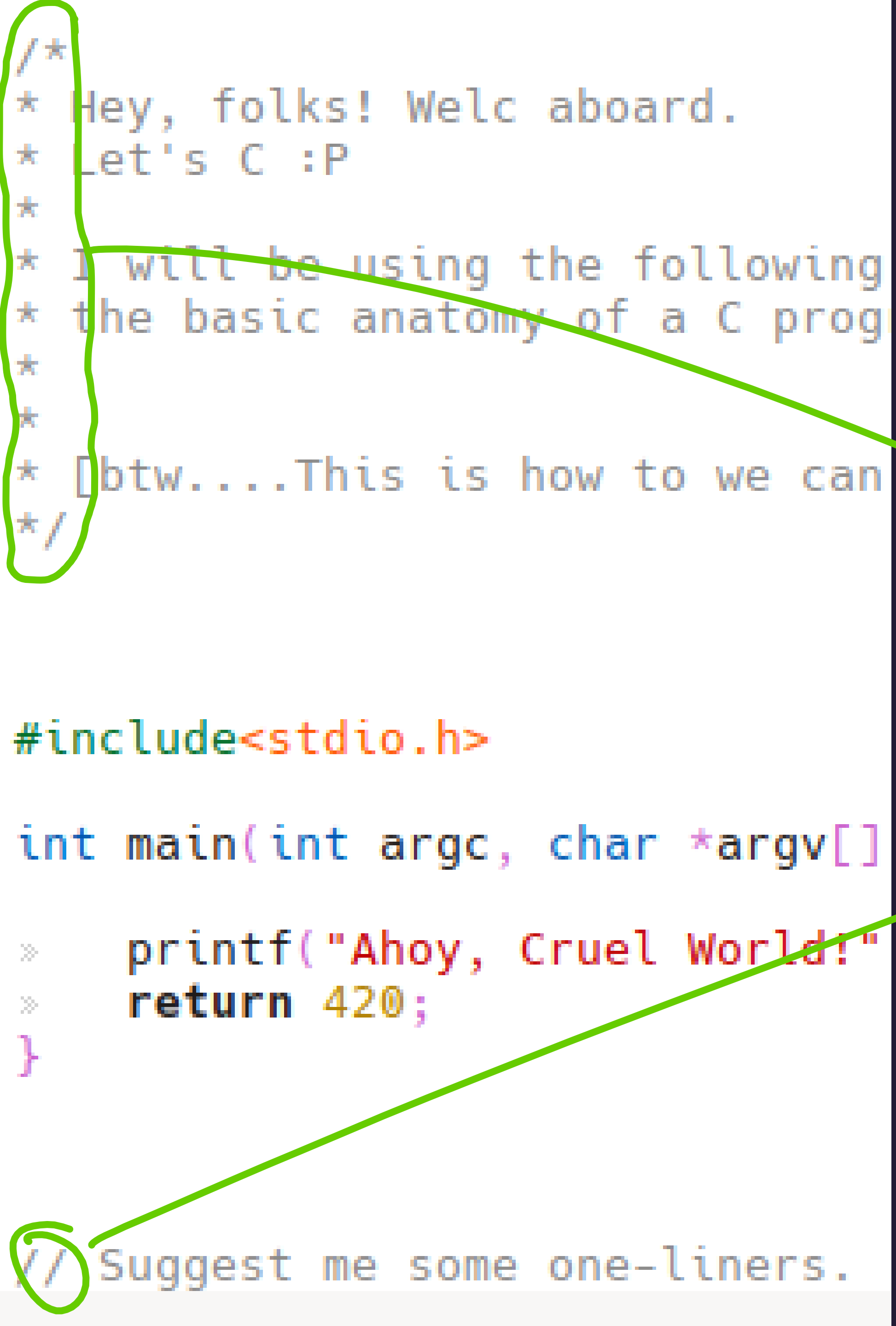
- C as a procedural, general-purpose programming language.
- Its efficient and portable.
- Mention its role as the foundation for many other languages.



A BRIEF HISTORY OF C

- Created by Dennis Ritchie at Bell Labs in 1970s.
- Evolved from BCPL and B.
- Designed for UNIX OS.
- Known for efficiency and portability.
- Basis for C++, Java.
- Standardized in 1989.



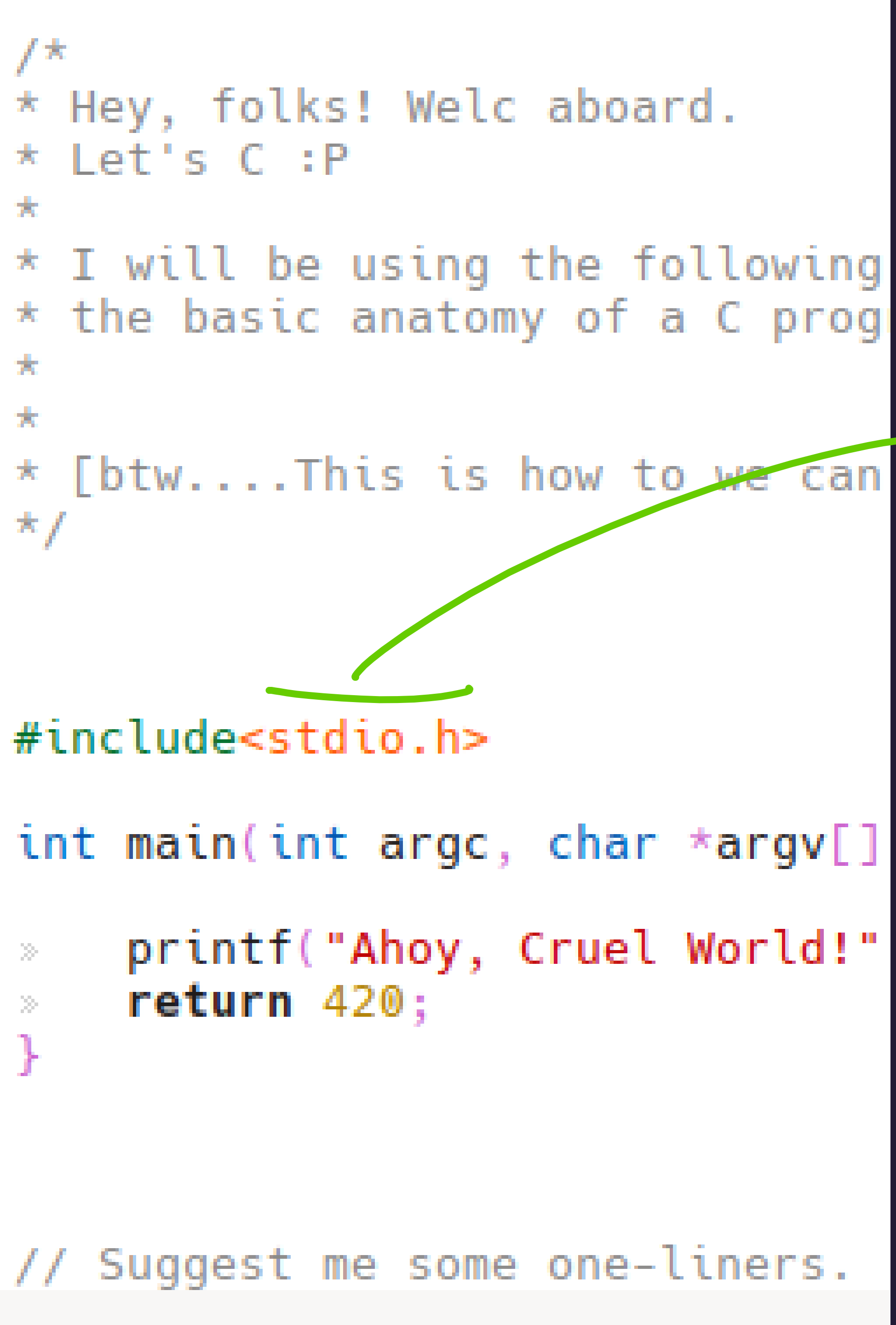


```
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* btw....This is how to we can
*/



#include<stdio.h>

int main(int argc, char *argv[]

    printf("Ahoy, Cruel World!"
    return 420;
}


// Suggest me some one-liners.
```

Anatomy

The compiler doesn't care!

We use this to provide info on It fun!

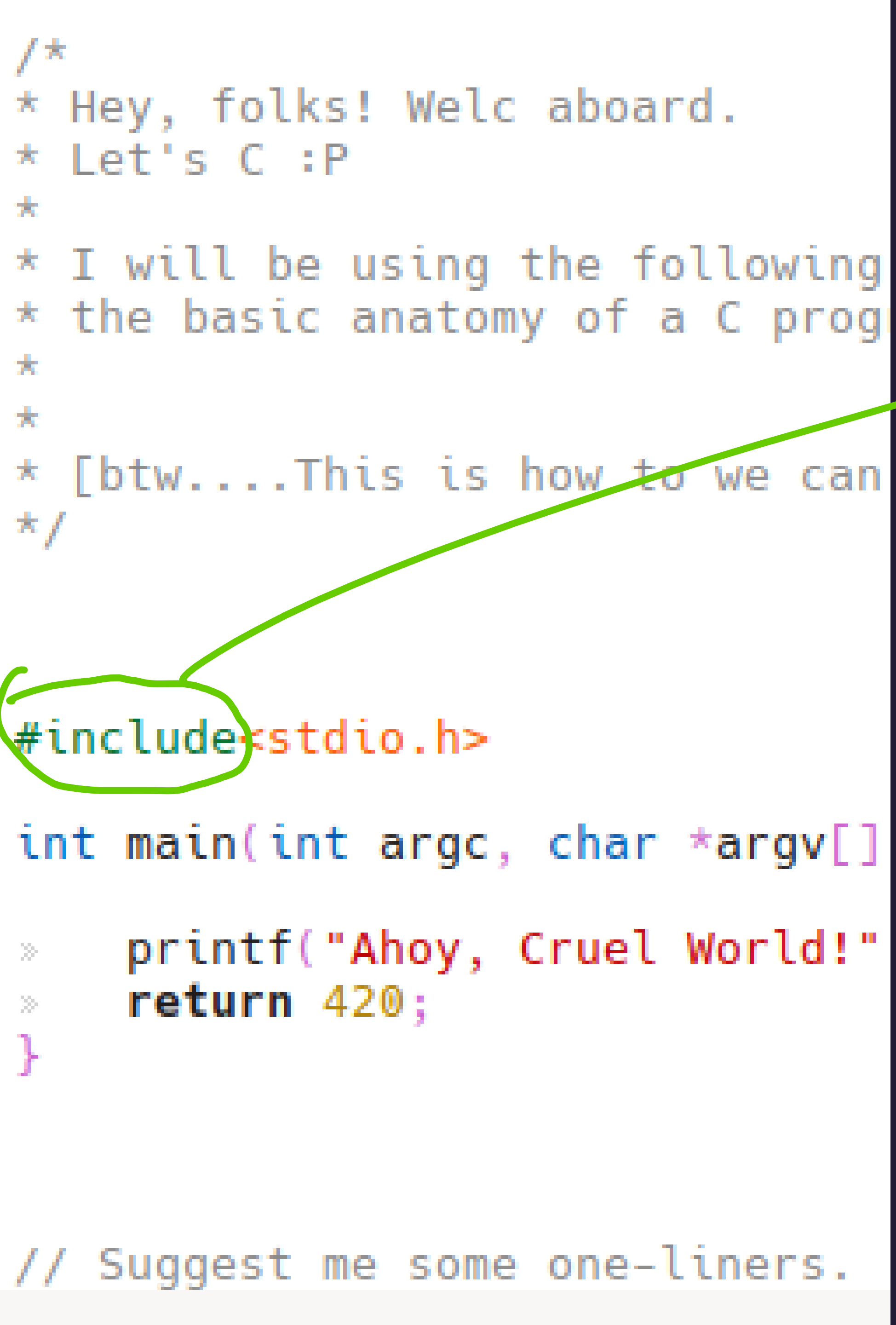


```c
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* [btw....This is how to we can
*/



#include<stdio.h>

int main(int argc, char *argv[]

	printf("Ahoy, Cruel World!"
	return 420;
}


// Suggest me some one-liners.
```

Anatomy

Header File

It stores a lotta' prewritten codes so that we can keep being lazy.



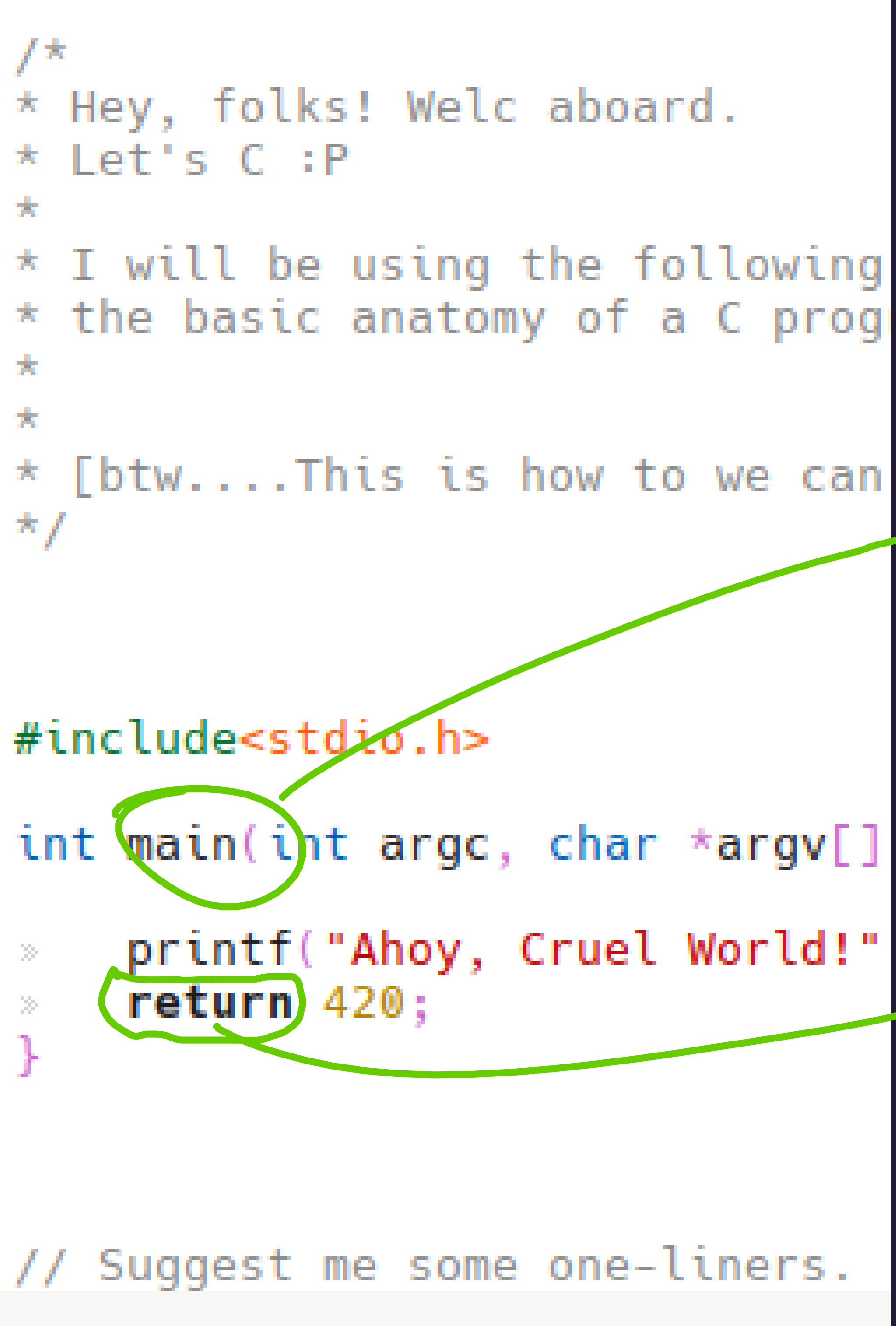
```c
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* [btw....This is how to we can
*/


#include<stdio.h>

int main(int argc, char *argv[]

»   printf("Ahoy, Cruel World!"
»   return 420;
}


// Suggest me some one-liners.
```

Anatomy

→ Preprocessor Directives.

i.e. "Hey' bro!"

'Include <this>'



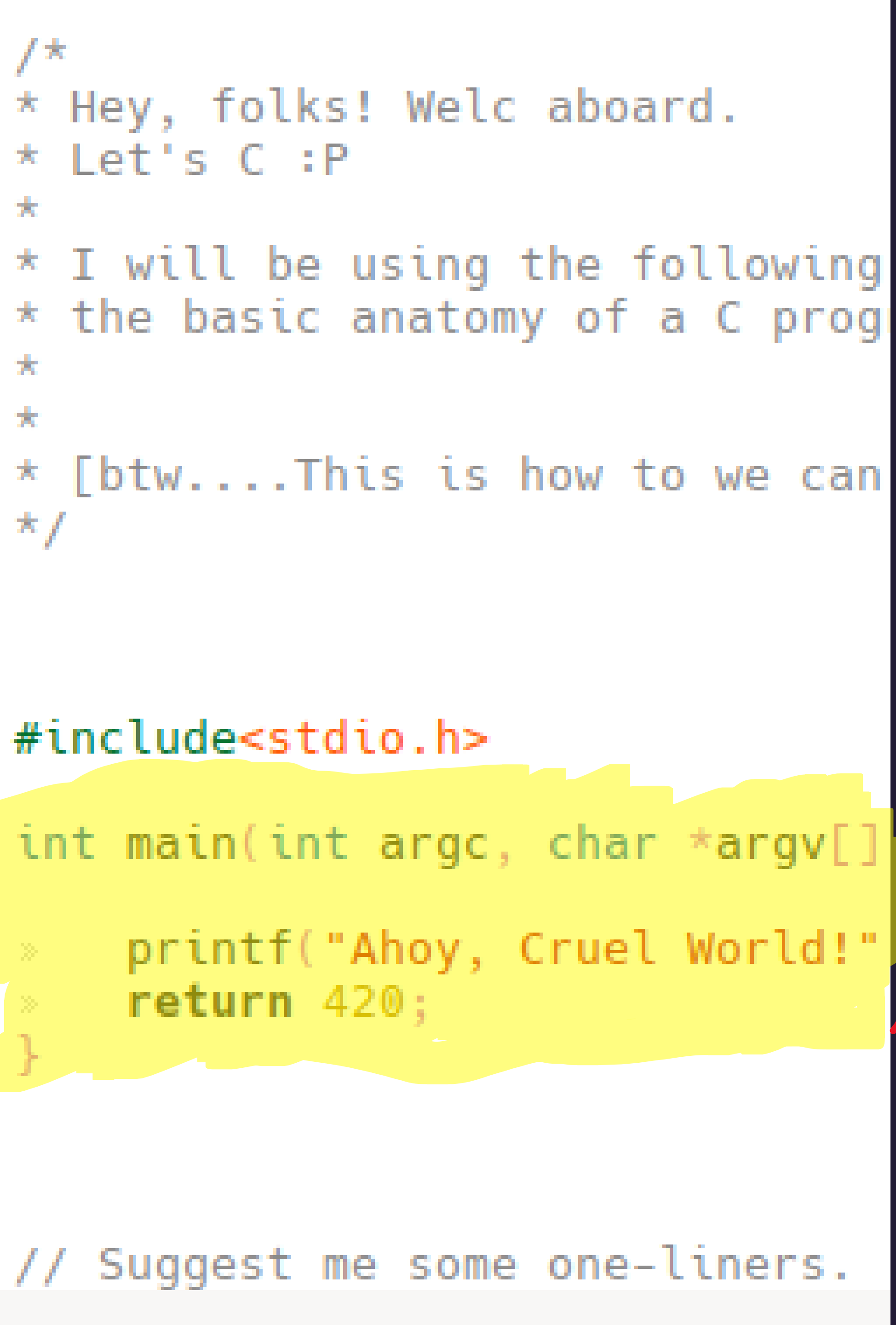
```
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* [btw....This is how to we can
*/



#include<stdio.h>

int main(int argc, char *argv[]
{
»   printf("Ahoy, Cruel World!"
»   return 420;
}


// Suggest me some one-liners.
```

Anatomy

The standard entry point.

exit

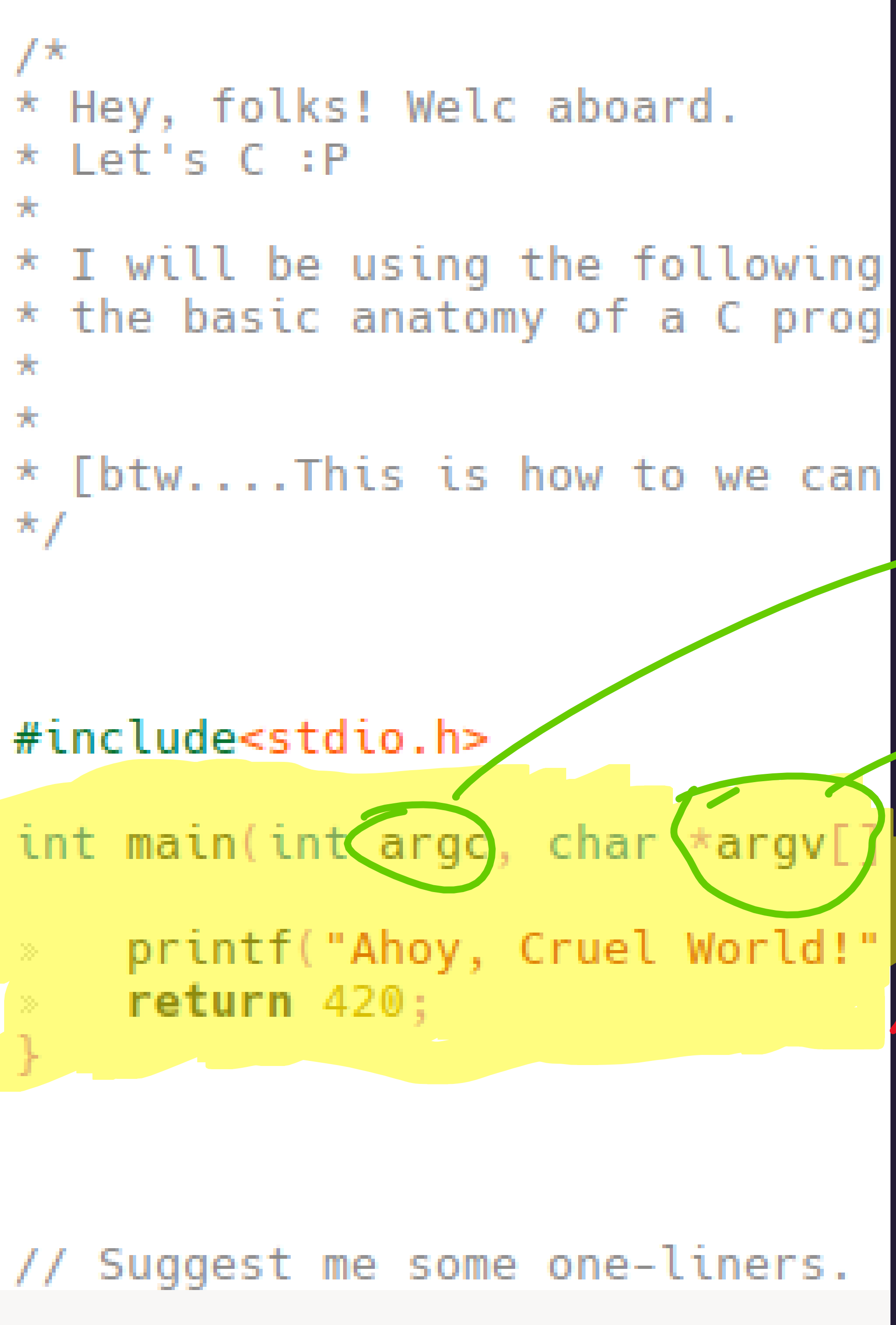
Anatomy



```c
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* [btw....This is how to we can
*/



#include<stdio.h>

int main(int argc, char *argv[]

    printf("Ahoy, Cruel World!"
    return 420;
}


// Suggest me some one-liners.
```

Actual Program

Anatomy

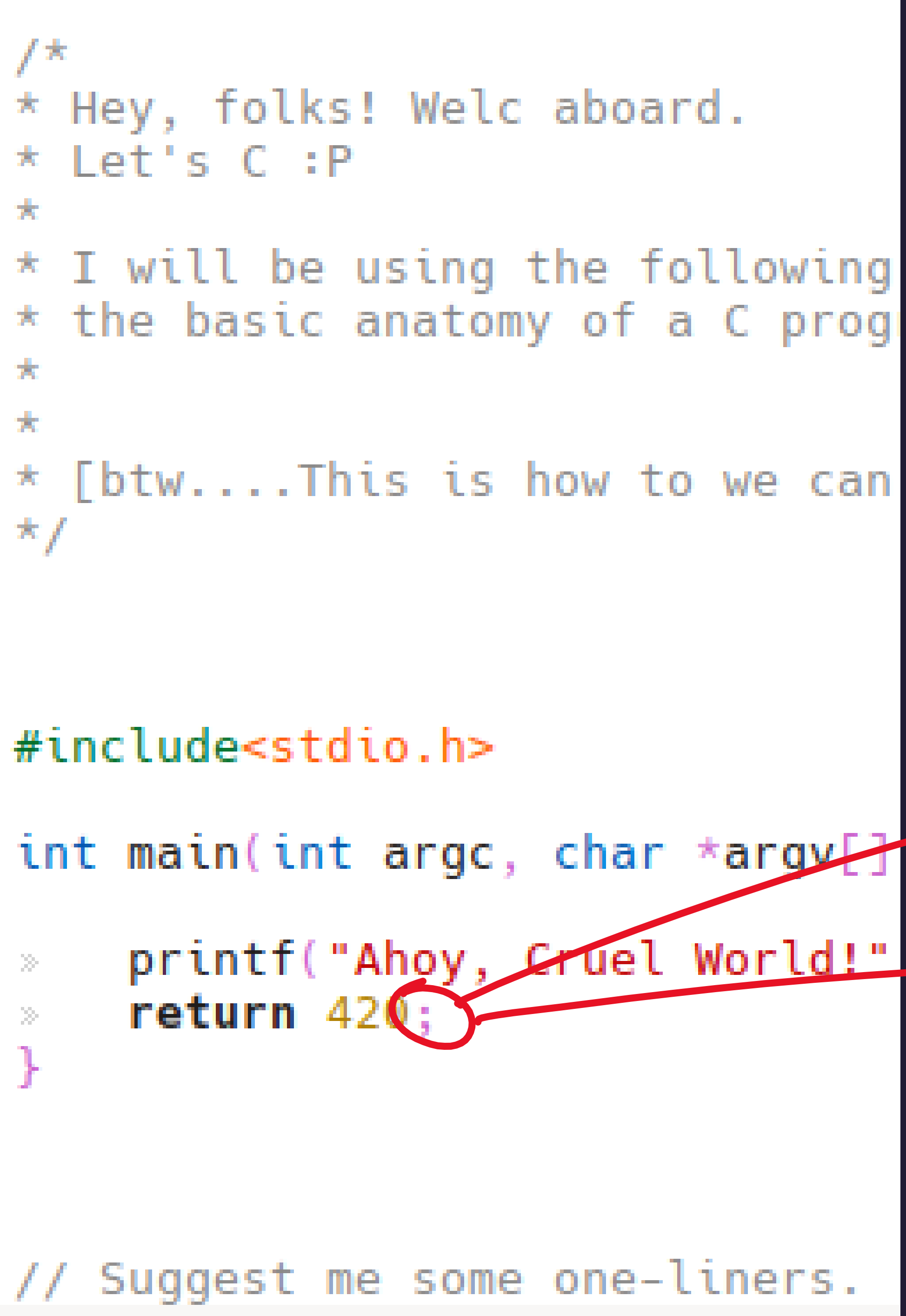


```
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* [btw....This is how to we can
*/



#include<stdio.h>

int main(int argc, char *argv[]

    printf("Ahoy, Cruel World!"
    return 420;
}


// Suggest me some one-liners.
```

Tracks # char strings passed

Whatever We pass

Actual Program

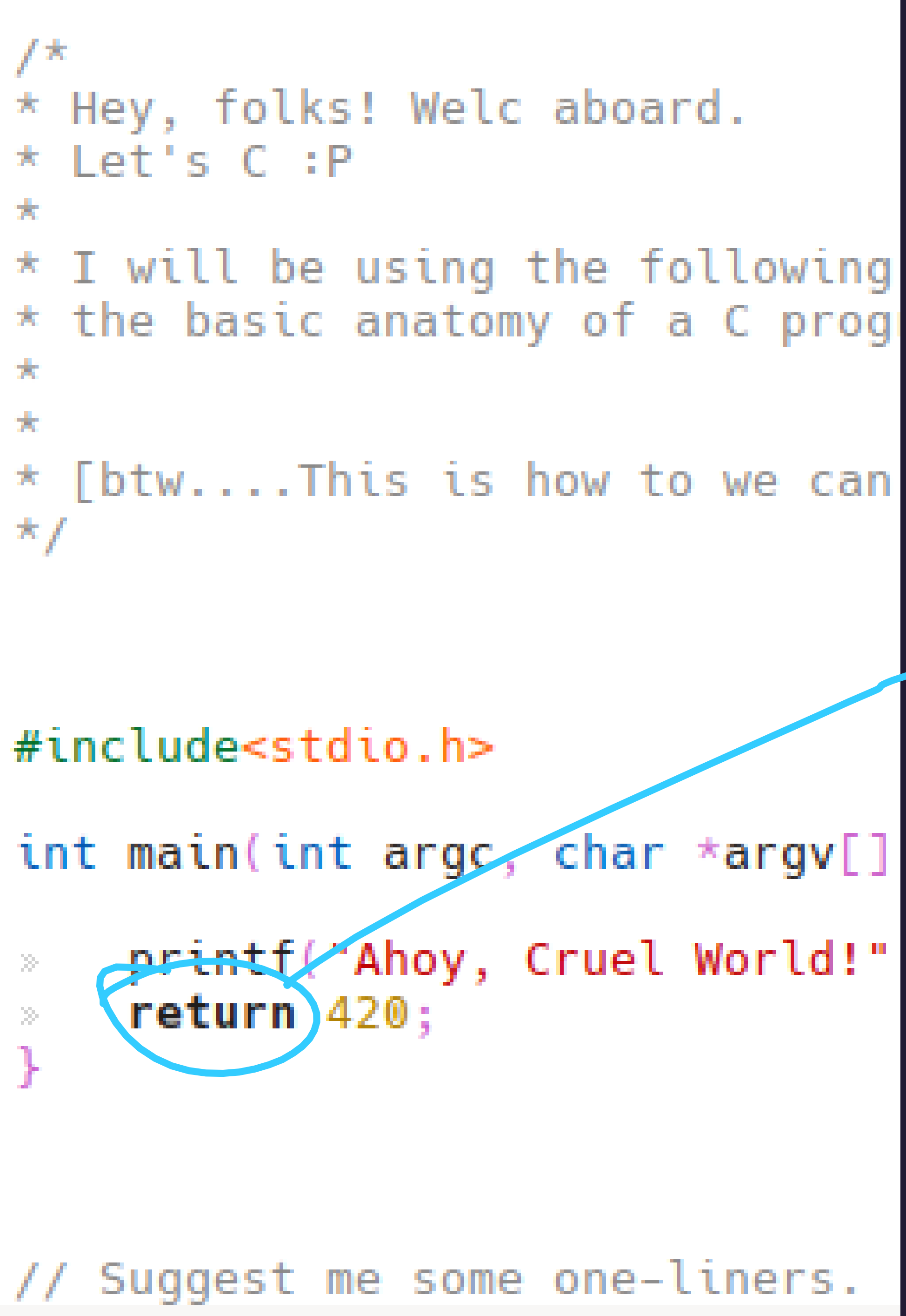


```
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* [btw....This is how to we can
*/



#include<stdio.h>

int main(int argc, char *argv[]

»   printf("Ahoy, Cruel World!"
»   return 420;
}


// Suggest me some one-liners.
```

Anatomy

Collumn/Sep

Liner



```
/*
* Hey, folks! Welc aboard.
* Let's C :P
*
* I will be using the following
* the basic anatomy of a C prog
*
*
* [btw....This is how to we can
*/


#include<stdio.h>

int main(int argc, char *argv[]

    printf("Ahoy, Cruel World!"
    return 420;
}


// Suggest me some one-liners.
```

Anatomy

But

where?!!!

C's Anatomy - Key Components

- **Comments:** Non-executable code for explanations.
- **Preprocessor directives:** Instructions for the compiler (e.g., #include).
- **Main function:** The program's entry point.
- **Statements:** Instructions executed sequentially.
- **Function calls:** Using pre-defined or user-defined functions.

C's Anatomy - Structure

- Starts with #include for necessary libraries.
- main function is the program's entry point.
- Code within curly braces defines the main function's body.
- Statements end with a semicolon.
- Program ends with a return statement.

C's Anatomy – Additional Notes

- Arguments can be passed to the main function for command-line input.
- Pointers are used to manipulate memory addresses.
- Indentation improves code readability.