EC-ENG 231 (Spring 2024)

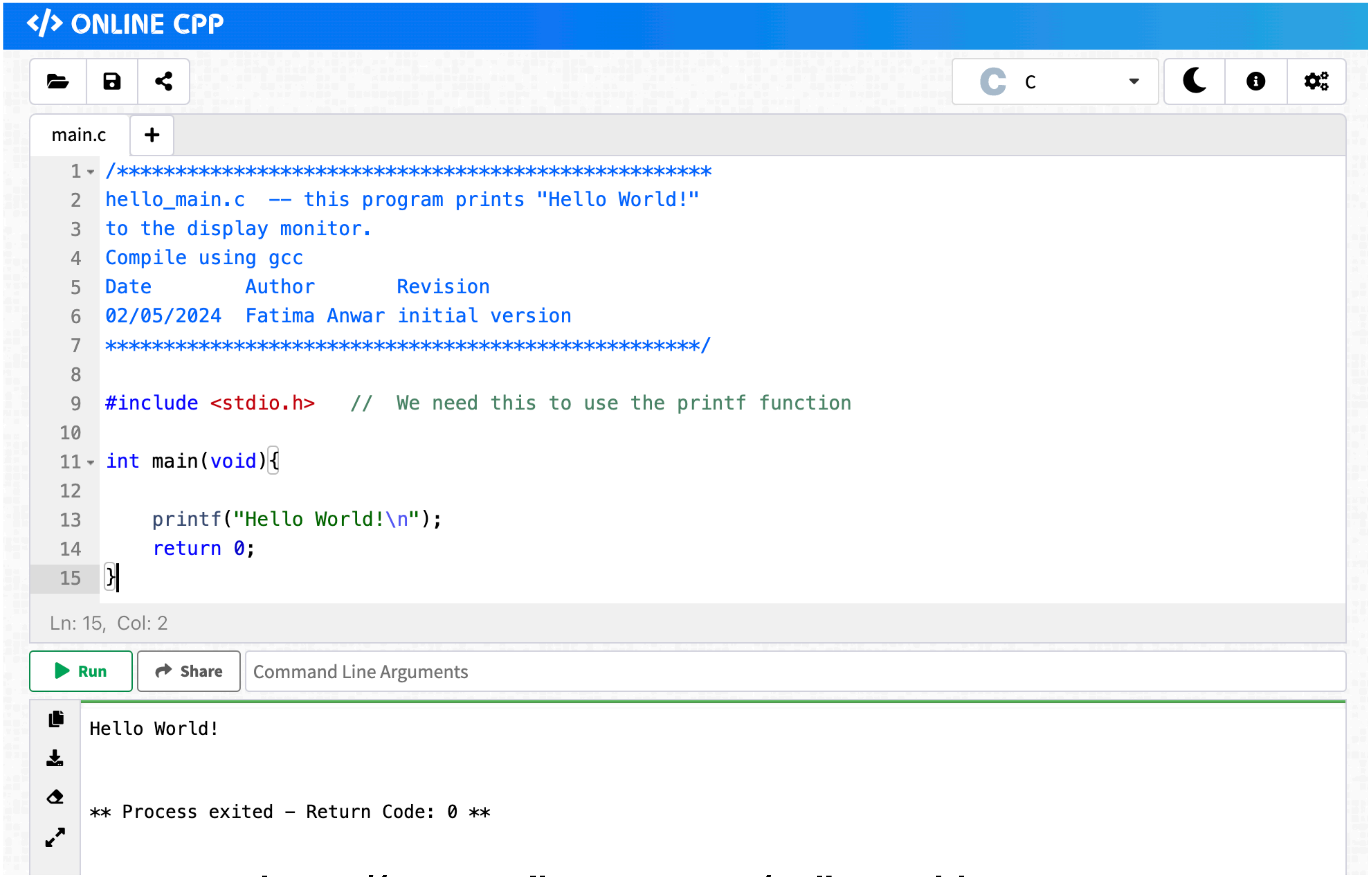
C Programming Part 2

Fatima Anwar

fanwar@umass.edu

UMass Amherst

Online C Compiler



```c
/***************************************************
hello_main.c  -- this program prints "Hello World!"
to the display monitor.
Compile using gcc
Date         Author       Revision
02/05/2024  Fatima Anwar initial version
***************************************************/

#include <stdio.h>    //  We need this to use the printf function

int main(void){

    printf("Hello World!\n");
    return 0;
}
```

```
Hello World!


** Process exited - Return Code: 0 **
```

https://www.online-cpp.com/online_c_ide

Control Structures

Conditionals
Loops
Statement

Control Structures

- if
- if-else
- for loop
- while loop
- do while loop
- switch statement

if block

0 is false
non-zero is true

```
if (condition)
(statement)
```

single statement doesn't require braces. But most style guides recommend them.

```
if (condition)
{
      ...
}
```

```
if (condition){
   ...
}
```

curly braces vertically aligned; easier to read.

first curly brace in line with condition; saves space but harder to read.

which style to use? Either. Pick one and be consistent with it in your coding.

if/else control structure

```
if (condition)
{
        ...
}
else
{
        ...
}
```

curly braces vertically aligned
– easier to read.

```
if (condition){
   ...
} else {
   ...
}
```

first curly brace in line with condition.
Saves space but harder braces to read.

which style to use? Either. Pick one and be
consistent with it in your coding.

Here’s the mistake many people make:

```
a = 10
b = 20
if (a=b) {
  print(“They’re equal!”);
} else {
  print(“They’re not equal\n)
}
```

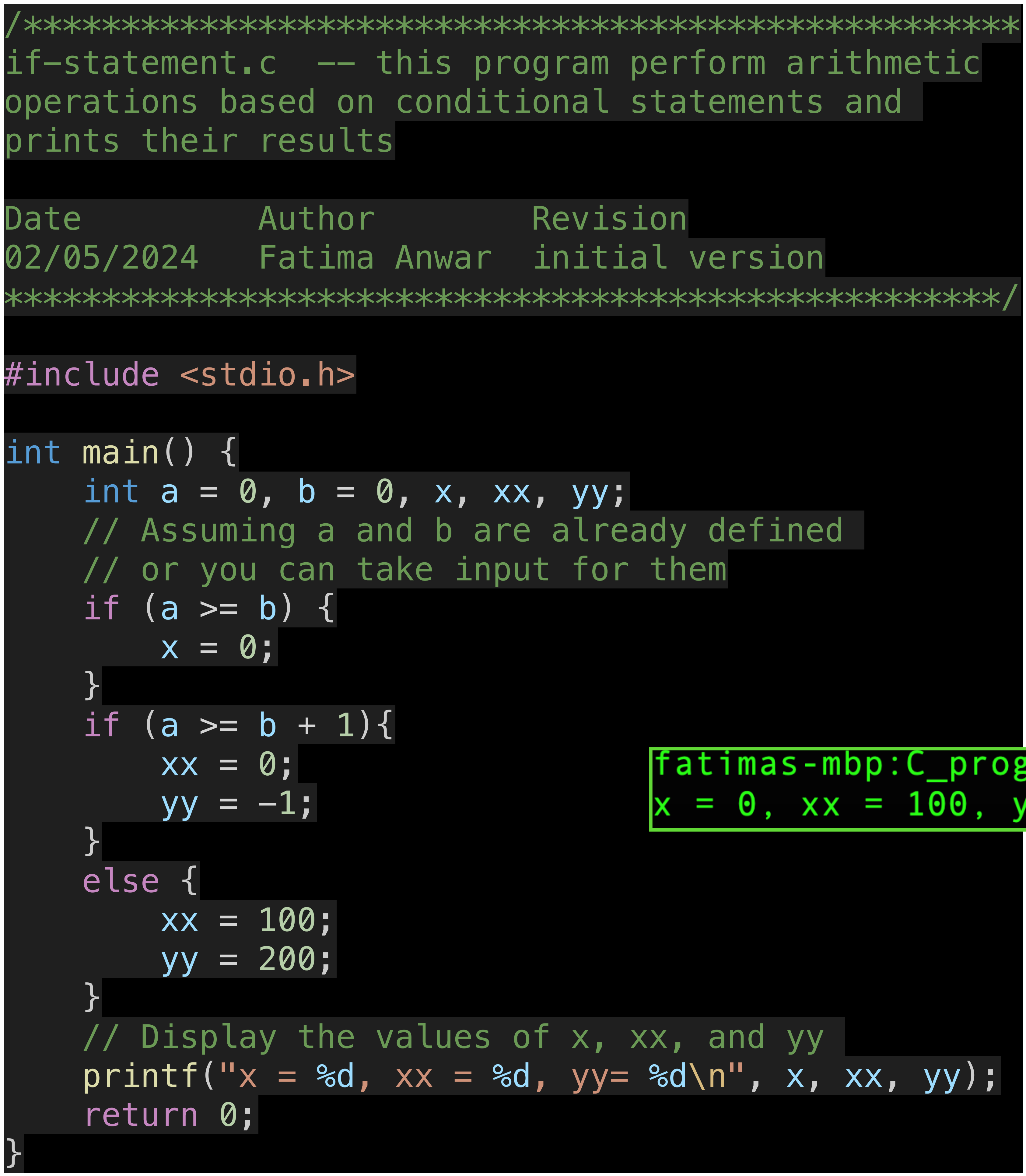
= is the assignment operator.
== tests for equality

C language philosophy: trust the programmer.
C will let you really screw up!

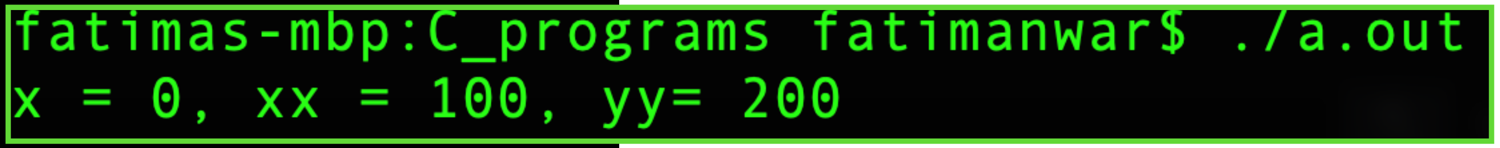
if/else-if/else control structure

```
if (condition1)
{
        ...
}
else if (condition2)
{
        ...
}
else
{
        ...
}
```

```
if (condition1){
    ...
} else if (condition2){
    ...
} else {
    ...
}
```



```c
/***************************************************
if-statement.c  -- this program perform arithmetic
operations based on conditional statements and
prints their results

Date          Author         Revision
02/05/2024    Fatima Anwar   initial version
***************************************************/

#include <stdio.h>

int main() {
    int a = 0, b = 0, x, xx, yy;
    // Assuming a and b are already defined
    // or you can take input for them
    if (a >= b) {
        x = 0;
    }
    if (a >= b + 1){
        xx = 0;
        yy = -1;
    }
    else {
        xx = 100;
        yy = 200;
    }
    // Display the values of x, xx, and yy
    printf("x = %d, xx = %d, yy= %d\n", x, xx, yy);
    return 0;
}
```



```
fatimas-mbp:C_programs fatimanwar$ ./a.out
x = 0, xx = 100, yy= 200
```

while loop

```
while (condition) {
     ...
}
```

keep looping as long condition is true (true means condition evaluates to a non-zero value)

```
example:
i = 0;
while (i < 10) {
     printf("This is iteration %d\n", i);
     i++;
}
```

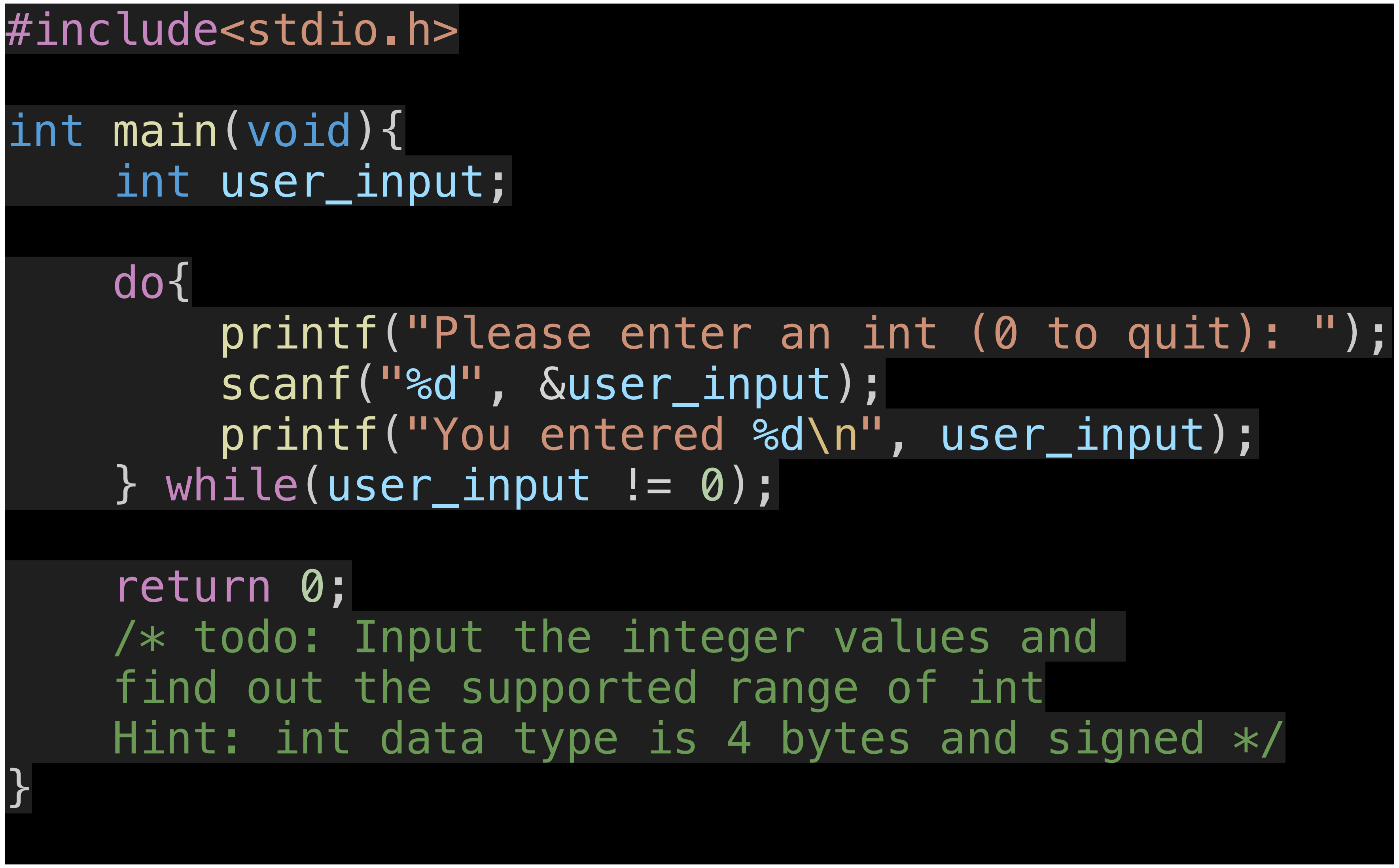
what happens if we don't have i++ ?
while (1) { ;} ? Infinite loops

do loop

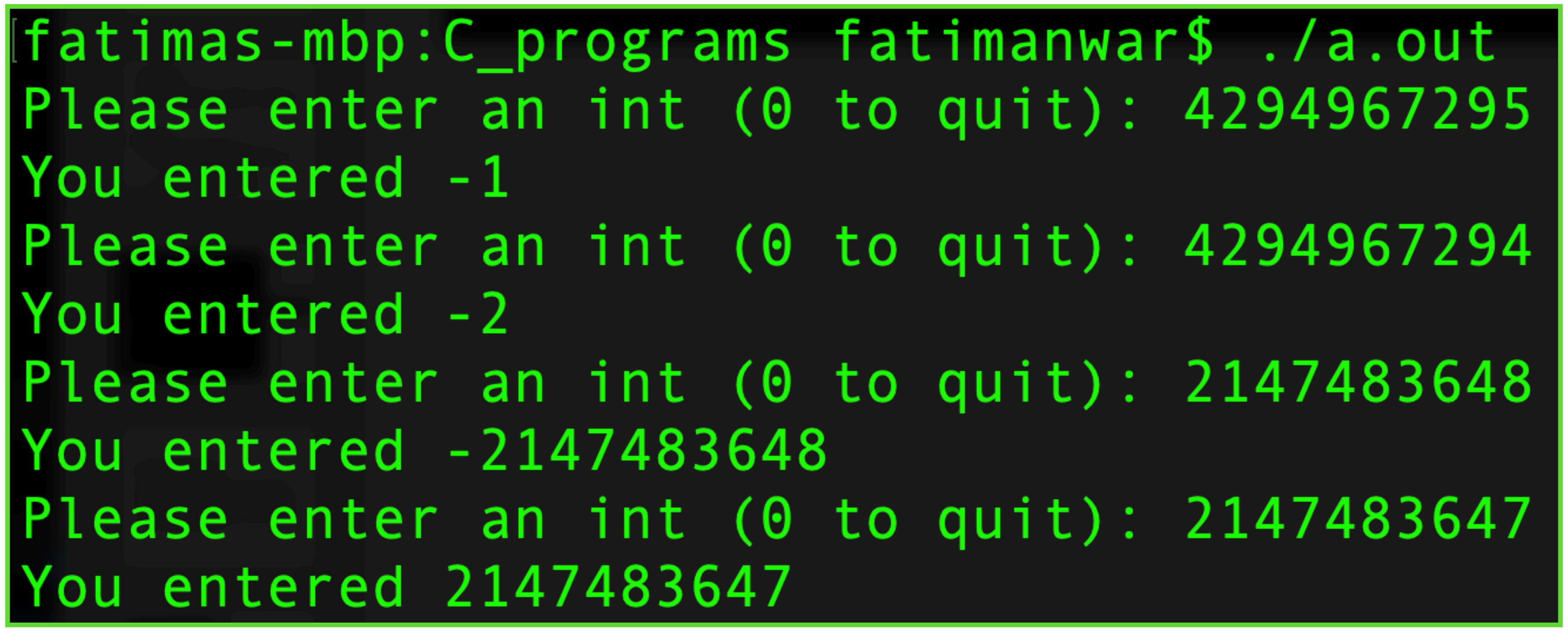
```
do {
     ...
} while (condition)

example:
i = 0;
do {
     printf("This is iteration %d\n", i);
     i++;
} while (i < 10)
```

keep looping as long condition is true (true means condition evaluates to a non-zero value)

This structure will always execute the {...} block at least once.



```c
#include<stdio.h>

int main(void){
    int user_input;

    do{
        printf("Please enter an int (0 to quit): ");
        scanf("%d", &user_input);
        printf("You entered %d\n", user_input);
    } while(user_input != 0);

    return 0;
    /* todo: Input the integer values and
    find out the supported range of int
    Hint: int data type is 4 bytes and signed */
}
```



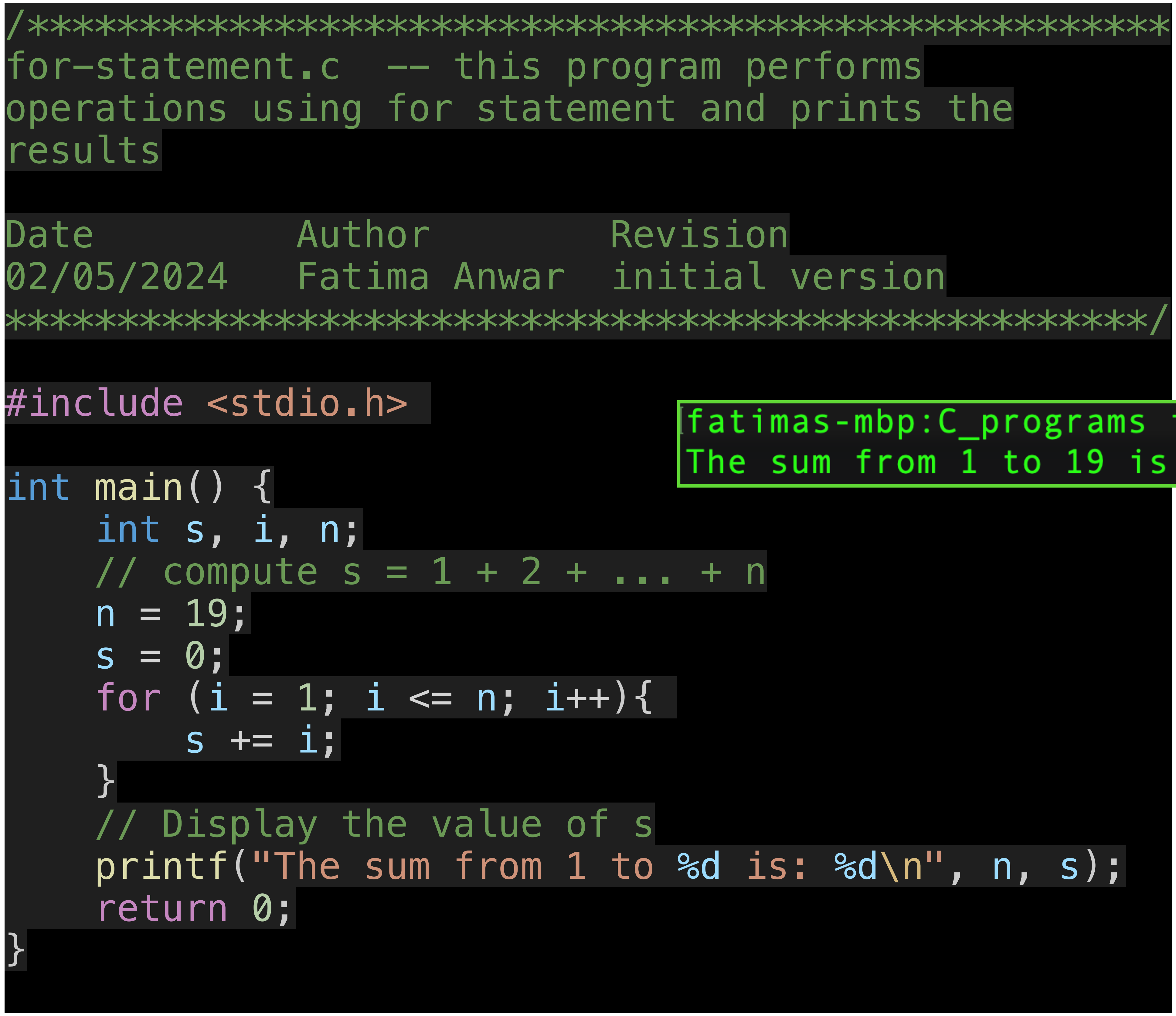
```
fatimas-mbp:C_programs fatimanwar$ ./a.out
Please enter an int (0 to quit): 4294967295
You entered -1
Please enter an int (0 to quit): 4294967294
You entered -2
Please enter an int (0 to quit): 2147483648
You entered -2147483648
Please enter an int (0 to quit): 2147483647
You entered 2147483647
```

for loop

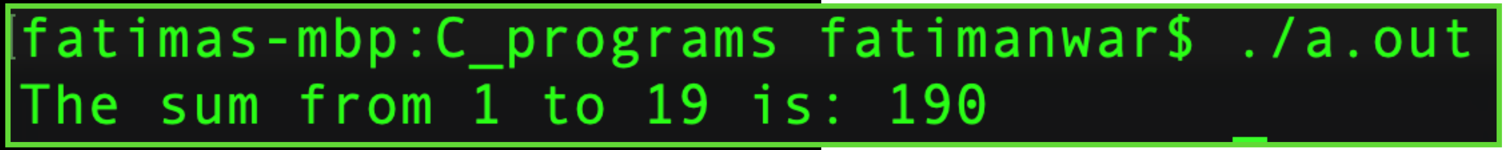
```
for (initialization; condition; update) {
    ...
}

example:
for (i = 1; i < 10; i++) {
    printf(“This is iteration %d\n”, i);
}
```

initialization, condition, update are all optional !

for(; ;) { ; } ?   Infinite loop



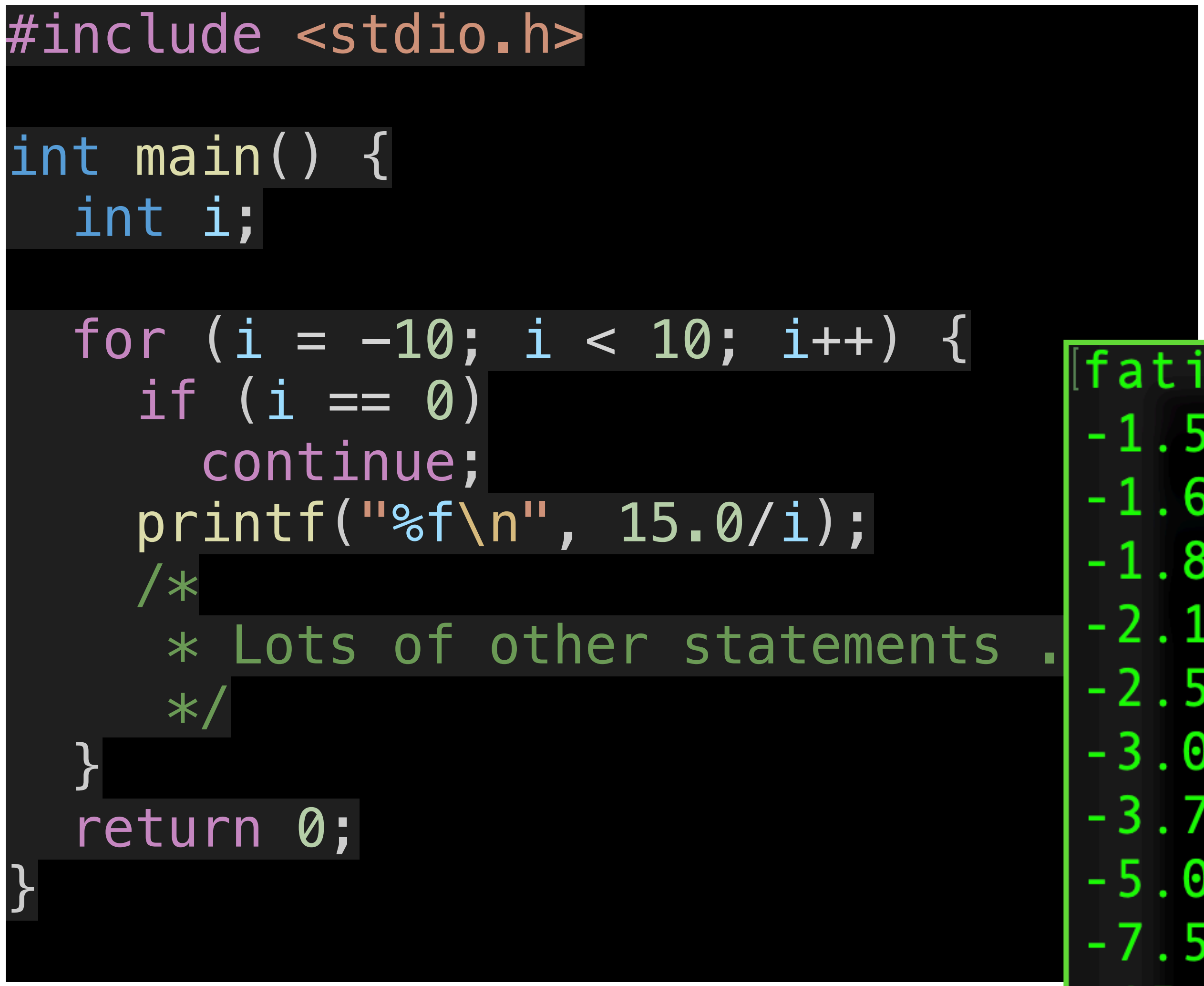
```c
/**************************************************
for-statement.c  -- this program performs
operations using for statement and prints the
results

Date          Author         Revision
02/05/2024    Fatima Anwar   initial version
**************************************************/

#include <stdio.h>

int main() {
    int s, i, n;
    // compute s = 1 + 2 + ... + n
    n = 19;
    s = 0;
    for (i = 1; i <= n; i++){
        s += i;
    }
    // Display the value of s
    printf("The sum from 1 to %d is: %d\n", n, s);
    return 0;
}
```



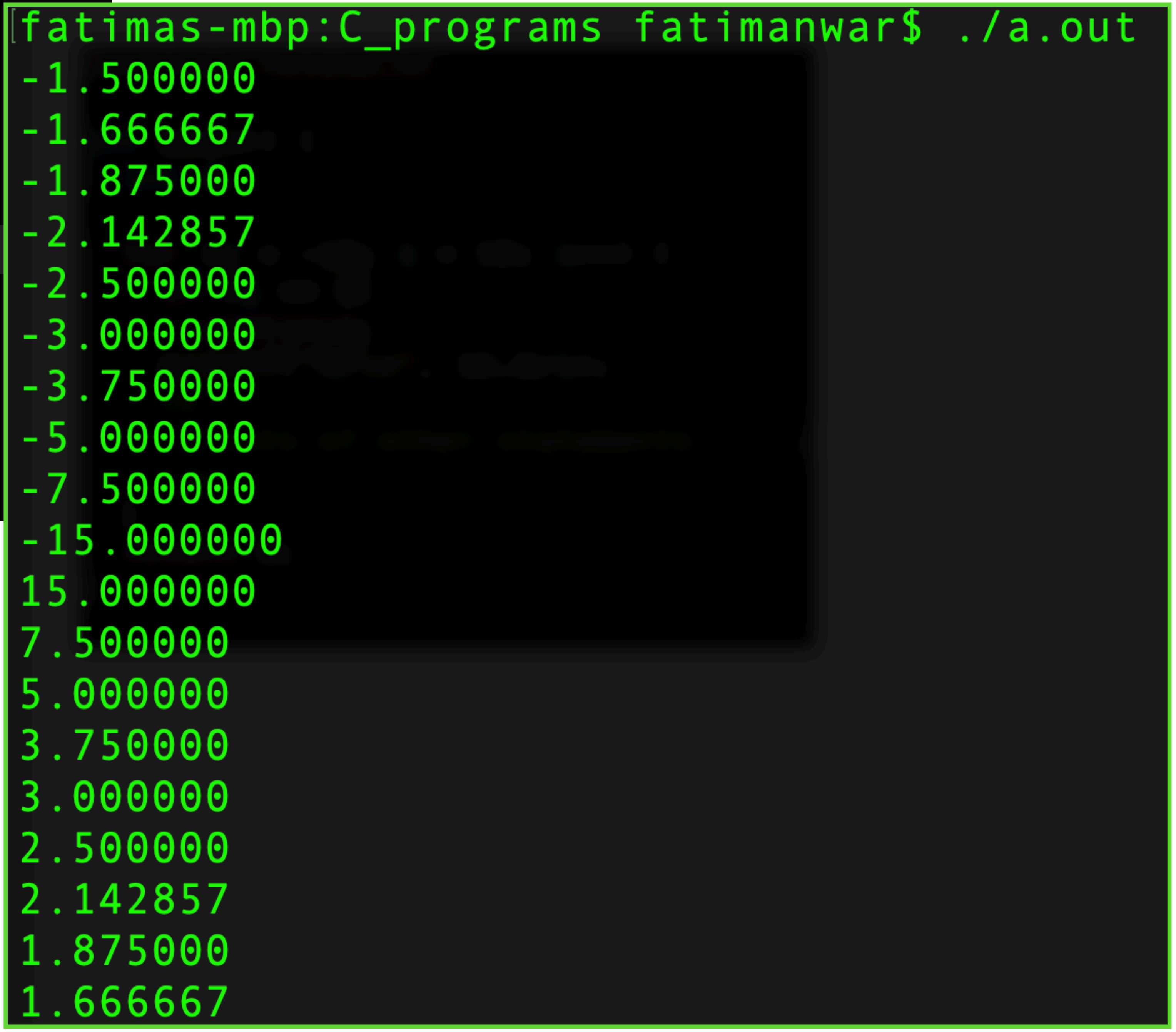
```
fatimas-mbp:C_programs fatimanwar$ ./a.out
The sum from 1 to 19 is: 190
```

break & continue

- break – control is exited from the construct (loop, struct) immediately
- continue – control is passed to the beginning of the construct (loop statement)

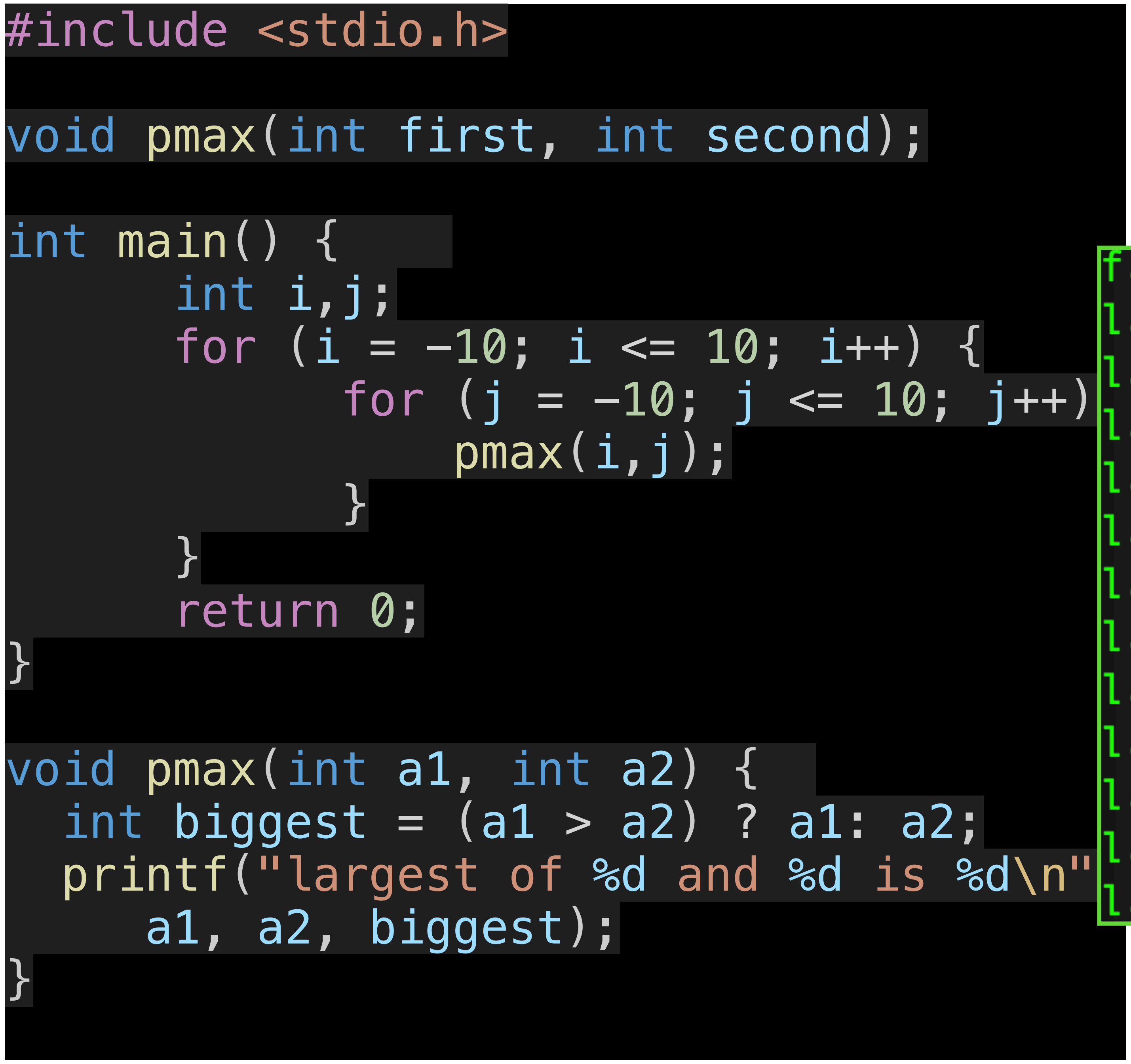


```
#include <stdio.h>

int main() {
   int i;

   for (i = -10; i < 10; i++) {
     if (i == 0)
       continue;
     printf("%f\n", 15.0/i);
     /*
      * Lots of other statements .
      */
   }
   return 0;
}
```

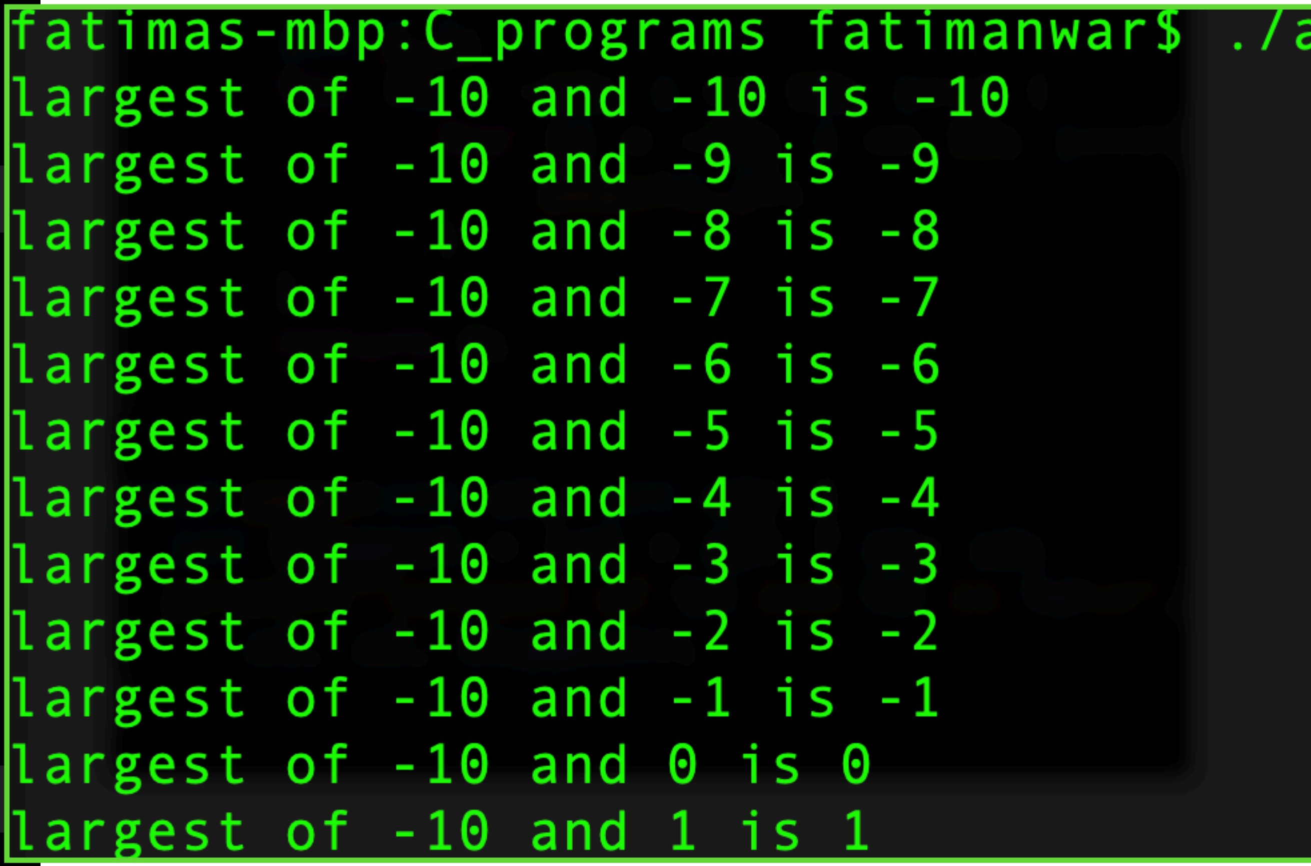


```
fatimas-mbp:C_programs fatimanwar$ ./a.out
-1.500000
-1.666667
-1.875000
-2.142857
-2.500000
-3.000000
-3.750000
-5.000000
-7.500000
-15.000000
15.000000
7.500000
5.000000
3.750000
3.000000
2.500000
2.142857
1.875000
1.666667
```

Nested for loops and functions



```c
#include <stdio.h>

void pmax(int first, int second);

int main() {
	int i,j;
	for (i = -10; i <= 10; i++) {
		for (j = -10; j <= 10; j++)
			pmax(i,j);
		}
	}
	return 0;
}

void pmax(int a1, int a2) {
  int biggest = (a1 > a2) ? a1: a2;
  printf("largest of %d and %d is %d\n"
     a1, a2, biggest);
}
```



```
fatimas-mbp:C_programs fatimanwar$ ./a
largest of -10 and -10 is -10
largest of -10 and -9 is -9
largest of -10 and -8 is -8
largest of -10 and -7 is -7
largest of -10 and -6 is -6
largest of -10 and -5 is -5
largest of -10 and -4 is -4
largest of -10 and -3 is -3
largest of -10 and -2 is -2
largest of -10 and -1 is -1
largest of -10 and 0 is 0
largest of -10 and 1 is 1
```

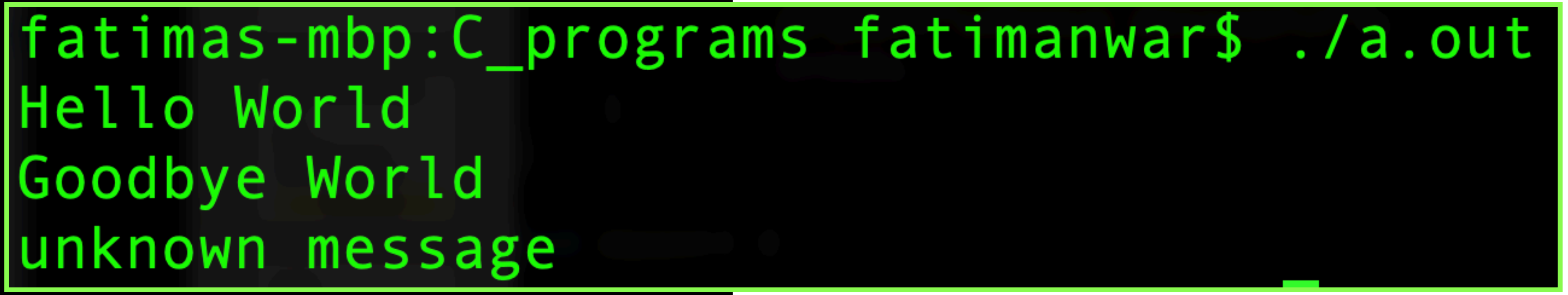
Switch statement

```
Switch (expression) { \\expression must be of type int, char
     case label_1:    \\if ‘label’ matches the expression,
          statement_1 \\execute this statement
          break
     case label_2:
          statement_2
          .
          .
          .
     default:
          statement_n
}

Example:
switch (letter){
  case ’N':
     printf("New York\n");
     break;
  default:
     printf("Somewhere else\n”);
     break;
}
```

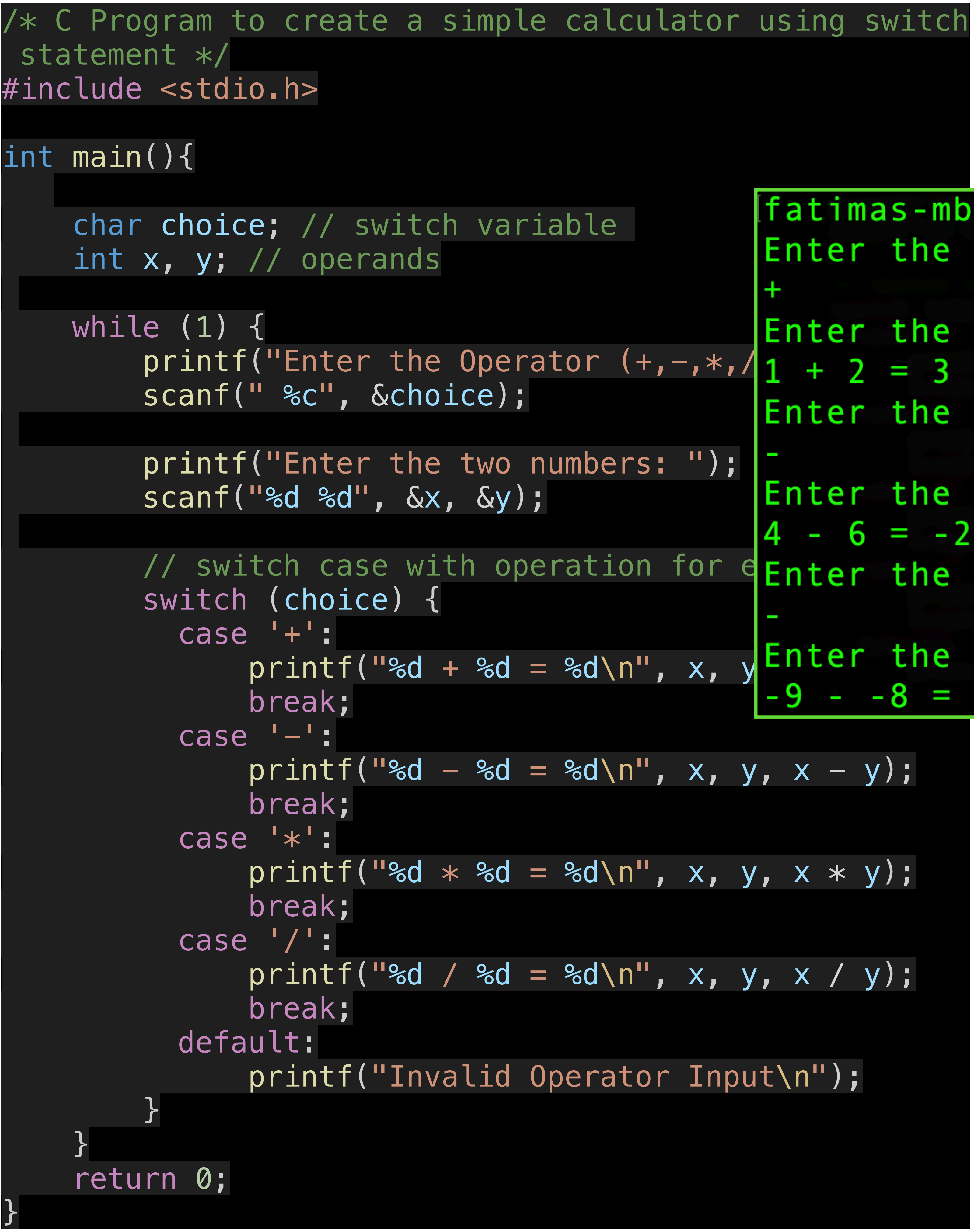
Notes:

- Substitute for long ‘if statements’
- Cases should be unique
- ‘break’ is optional
- ‘default’ case is optional

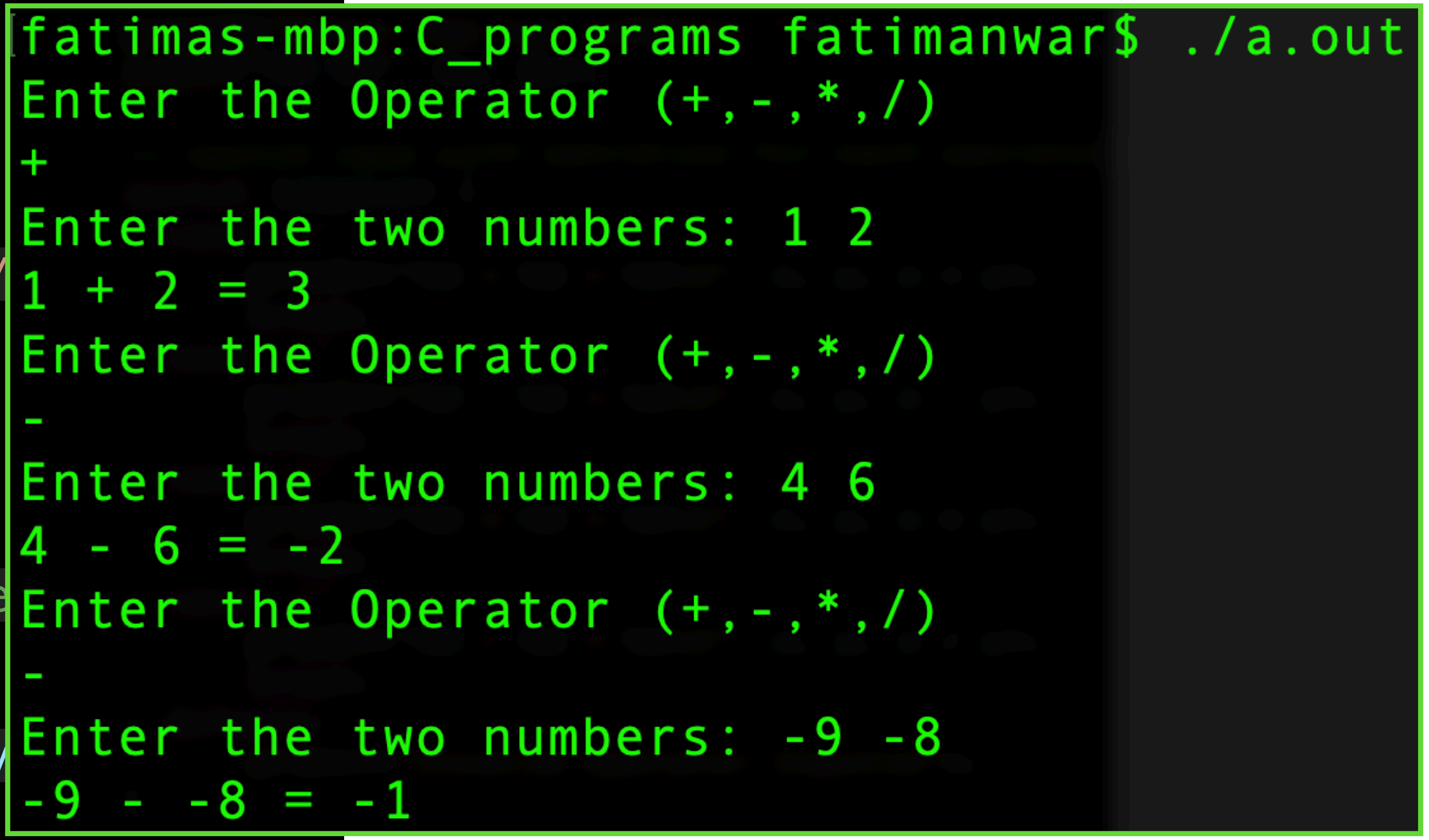
```c
/*****************************************************
Message printing function

Date           Author          Revision
02/05/2024     Fatima Anwar    initial version
*****************************************************/

#include <stdio.h>

void message(int message_number) {
    switch (message_number) {
        case 1:
            printf("Hello World\n");
            break;
        case 2:
            printf("Goodbye World\n");
            break;
        default:
            printf("unknown message\n");
            break;
    }
}

int main() {

    int x = 2;
    message(1); /* What will happen? */
    message(x);
    message(0);

    return 0;
}
```



```
fatimas-mbp:C_programs fatimanwar$ ./a.out
Hello World
Goodbye World
unknown message
```



```c
/* C Program to create a simple calculator using switch
 statement */
#include <stdio.h>

int main(){

    char choice; // switch variable
    int x, y; // operands

    while (1) {
        printf("Enter the Operator (+,-,*,/
        scanf(" %c", &choice);

        printf("Enter the two numbers: ");
        scanf("%d %d", &x, &y);

        // switch case with operation for e
        switch (choice) {
          case '+':
              printf("%d + %d = %d\n", x, y
              break;
          case '-':
              printf("%d - %d = %d\n", x, y, x - y);
              break;
          case '*':
              printf("%d * %d = %d\n", x, y, x * y);
              break;
          case '/':
              printf("%d / %d = %d\n", x, y, x / y);
              break;
          default:
              printf("Invalid Operator Input\n");
        }
    }
    return 0;
}
```



```
fatimas-mbp:C_programs fatimanwar$ ./a.out
Enter the Operator (+,-,*,/)
+
Enter the two numbers: 1 2
1 + 2 = 3
Enter the Operator (+,-,*,/)
-
Enter the two numbers: 4 6
4 - 6 = -2
Enter the Operator (+,-,*,/)
-
Enter the two numbers: -9 -8
-9 - -8 = -1
```

Structured C types

Arrays
Pointers
Structs
Strings

Arrays in C

array – a data structure that can store a fixed-size collection of elements of the same data type.

```
float  temps[5] = { 98.6, 100.2, 99.3, 100.4, 96.7};
```

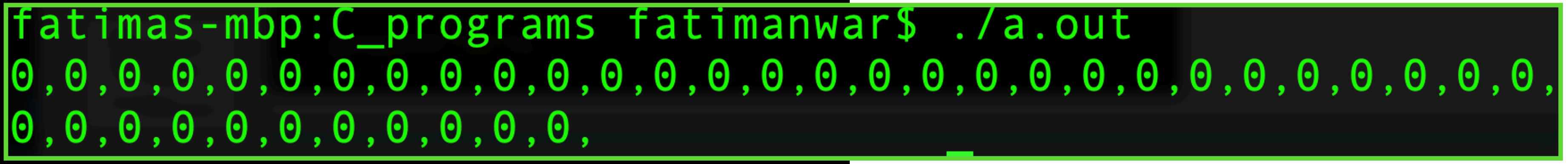
temp[0]　　　　temp[4]

```
float  temps[] = { 98.6, 100.2, 99.3, 100.4, 96.7};
```

The array index goes from 0 to arraysize -1

OK to leave the array size unspecified in the declaration. The compiler will figure it out.

```c
/* Function to initialize an array of integers
to a given initial value. Parameters passed:
a, the array of integers
n, number of elements to be initialized
val, the initial value */

#include <stdio.h>

//declare function
void initialize(int init_array[], int num, int init_val);

int main() {

    int x[40];

    // Initialize array values to zero
    initialize(x, 40, 0);

    return 0;
}

//define function
void initialize(int a[], int n, int val) {
    int i;
    for (i = 0; i < n; i++) {
        a[i] = val;
        printf("%d,", a[i]);
    }
    printf("\n");
}
```

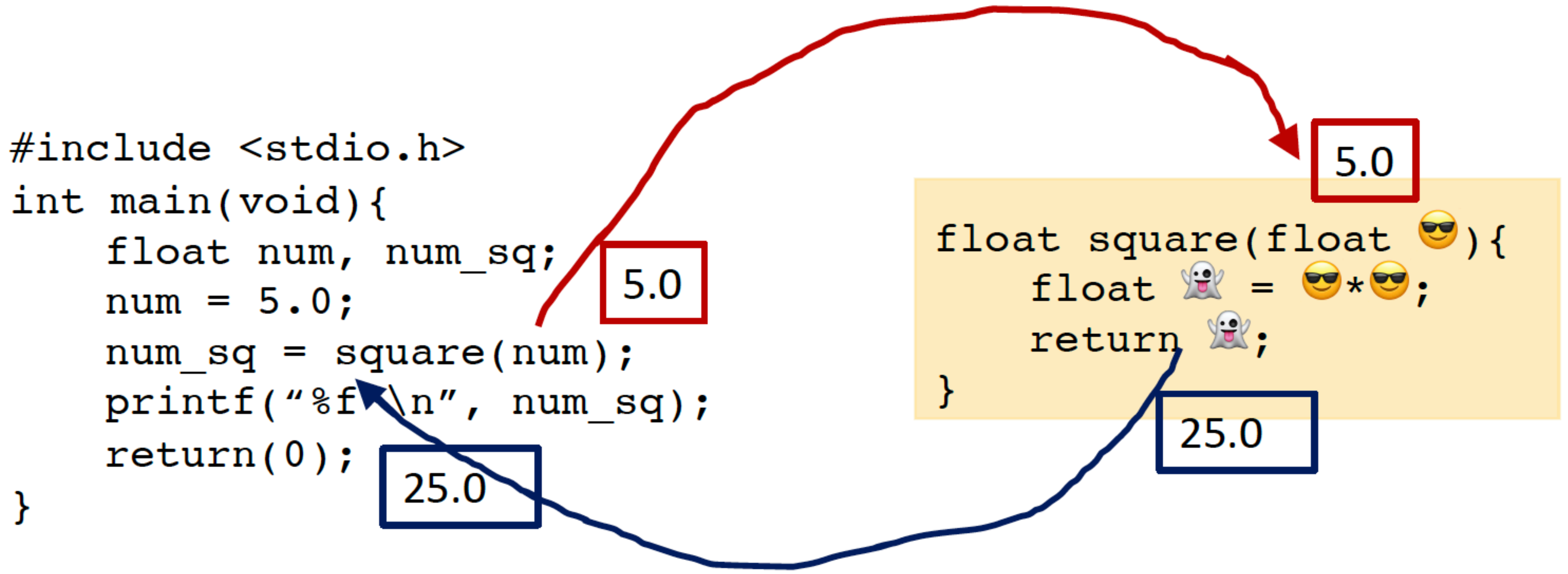


```
fatimas-mbp:C_programs fatimanwar$ ./a.out
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,
```

Scope: where a variable is visible

- Global scope – declared outside of a function; scope extends from the point of declaration to the end of the file in which it is declared
- Local scope – declared within a function; visible from the point of declaration to the end of the function body
- Statement scope – names declared in a block of code surrounded by {…} are only visible within that block. Applies to: for, if, while, switch statements

Pass by value



```
#include <stdio.h>
int main(void){
    float num, num_sq;
    num = 5.0;
    num_sq = square(num);
    printf("%f\n", num_sq);
    return(0);
}
```

```
float square(float 😎){
    float 👻 = 😎*😎;
    return 👻;
}
```

- The details of function square do not need to be known to main.
- ***Values*** of variables are passed in call & return.

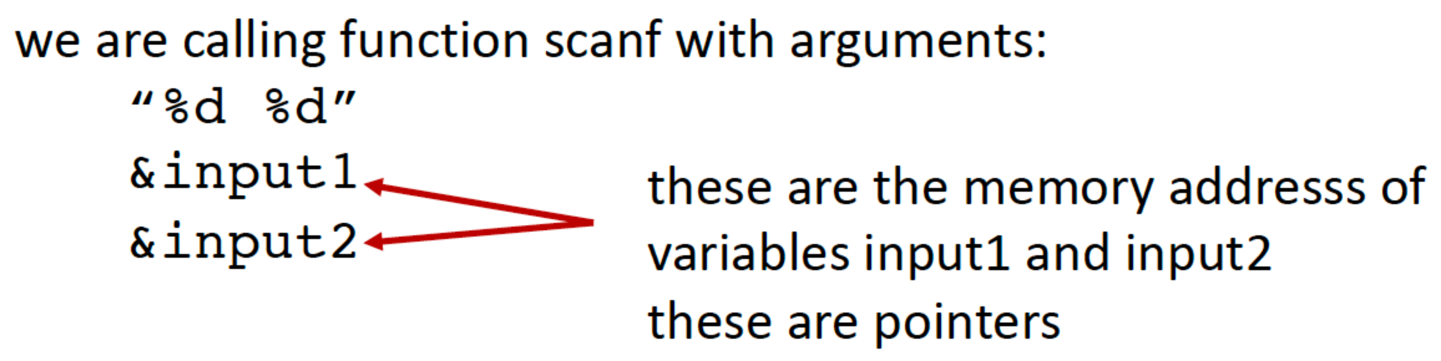
So, how can a function pass back more than one thing?
How can we "return" more than one thing?

Pointers

***** is used to declare pointers.
 It is also known as a dereferencing operator
 Accessing the object to which the pointer points is known as dereferencing

Note: First declare the pointer ***p.** Then assign a value to p. Use ***p** only after assigning a value to **p**.

Example: `scanf(“%d %d”, &input1, &input2)`



we are calling function scanf with arguments:

```
“%d %d”
&input1
&input2
```

these are the memory addresss of variables input1 and input2
these are pointers

“Pointers in C are easy and fun to learn.”

https://www.tutorialspoint.com/cprogramming/c_pointers.htm

Pointer Concepts

Key Concepts:

&a refers to the memory address of a
*p refers to the contents of memory address p

```
int a;        // a is an int
int *ptr_a;   // prt_a is a pointer to an int
a = 500;      // We know what this does.
ptr_a = &a;   // prt_a is the address of a
*ptr_a = 500;// the contents of mem location
              // ptr_a are set = 500
```

assign value to pointers before using them

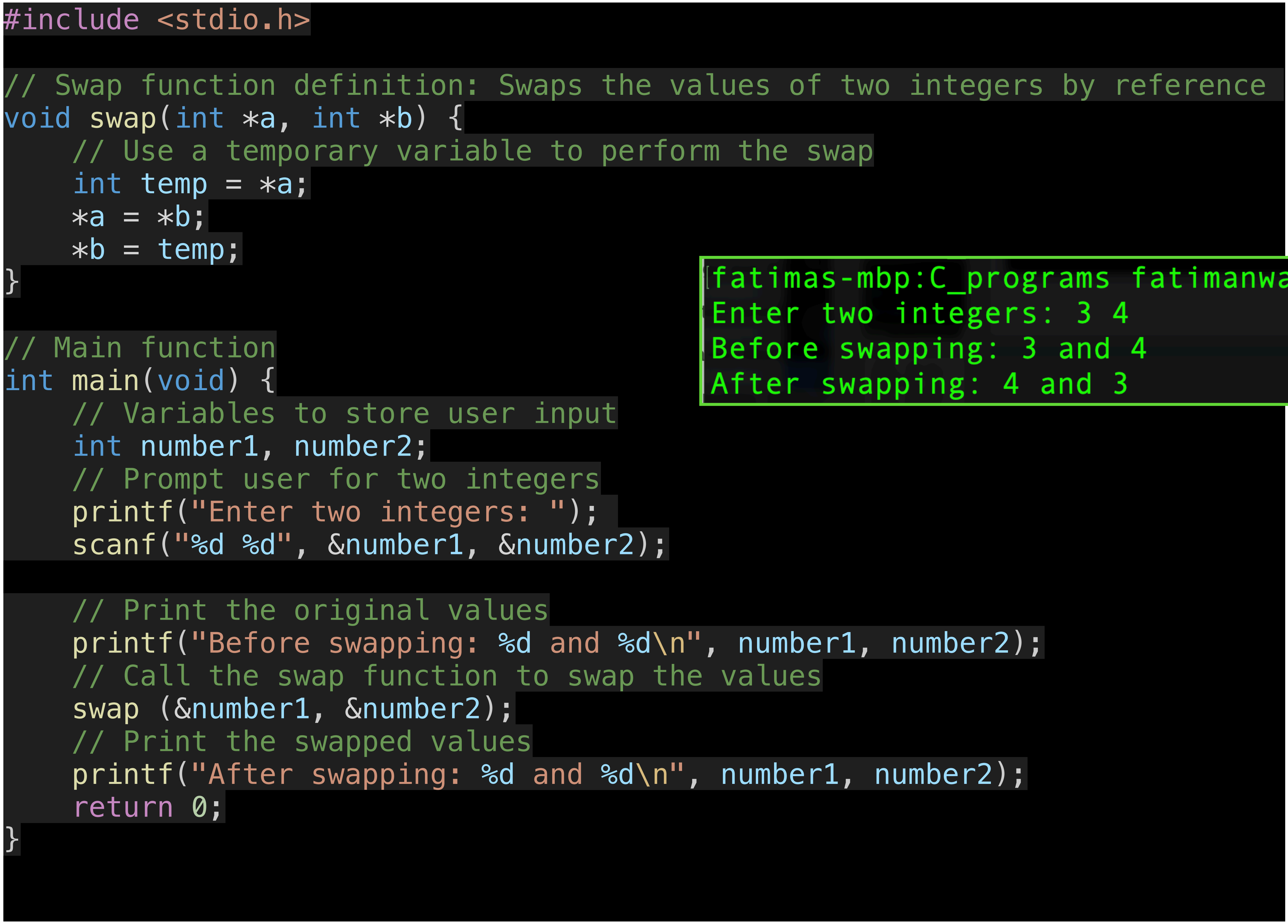
```c
#include <stdio.h>

int main(){
    char *p;
    char ch;
    p = &ch;
    *p = 'A';
    printf("%c\n", *p);
    return 0;
}
```

pointers to arrays

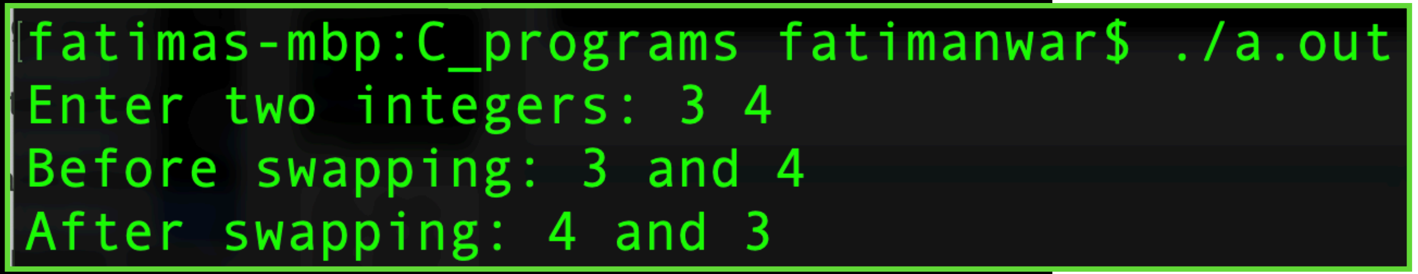
```c
#include <stdio.h>

int main() {

    // Declare and initialize an integer array
    int numbers[] = {1, 2, 3, 4, 5};

    // Declare a pointer to an integer
    int *ptr;
    // Point the pointer to the first element of the array
    ptr = numbers;

    // Modify the second element through the pointer
    *(ptr + 1) = 10;
    // Print the modified array
    for (int i = 0; i < 5; ++i) {
        printf("%d ", numbers[i]);
    }
    printf("\n");
    return 0;
}
```

- We pass the address of the first element of the array, which is the array name
- We automatically have access to all other elements in the array

parameter passing by reference using functions



```c
#include <stdio.h>

// Swap function definition: Swaps the values of two integers by reference
void swap(int *a, int *b) {
    // Use a temporary variable to perform the swap
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Main function
int main(void) {
    // Variables to store user input
    int number1, number2;
    // Prompt user for two integers
    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    // Print the original values
    printf("Before swapping: %d and %d\n", number1, number2);
    // Call the swap function to swap the values
    swap (&number1, &number2);
    // Print the swapped values
    printf("After swapping: %d and %d\n", number1, number2);
    return 0;
}
```



```
fatimas-mbp:C_programs fatimanwar$ ./a.out
Enter two integers: 3 4
Before swapping: 3 and 4
After swapping: 4 and 3
```

Struct

```
struct colour {
      int r;
      int g;
      int b;
};
```

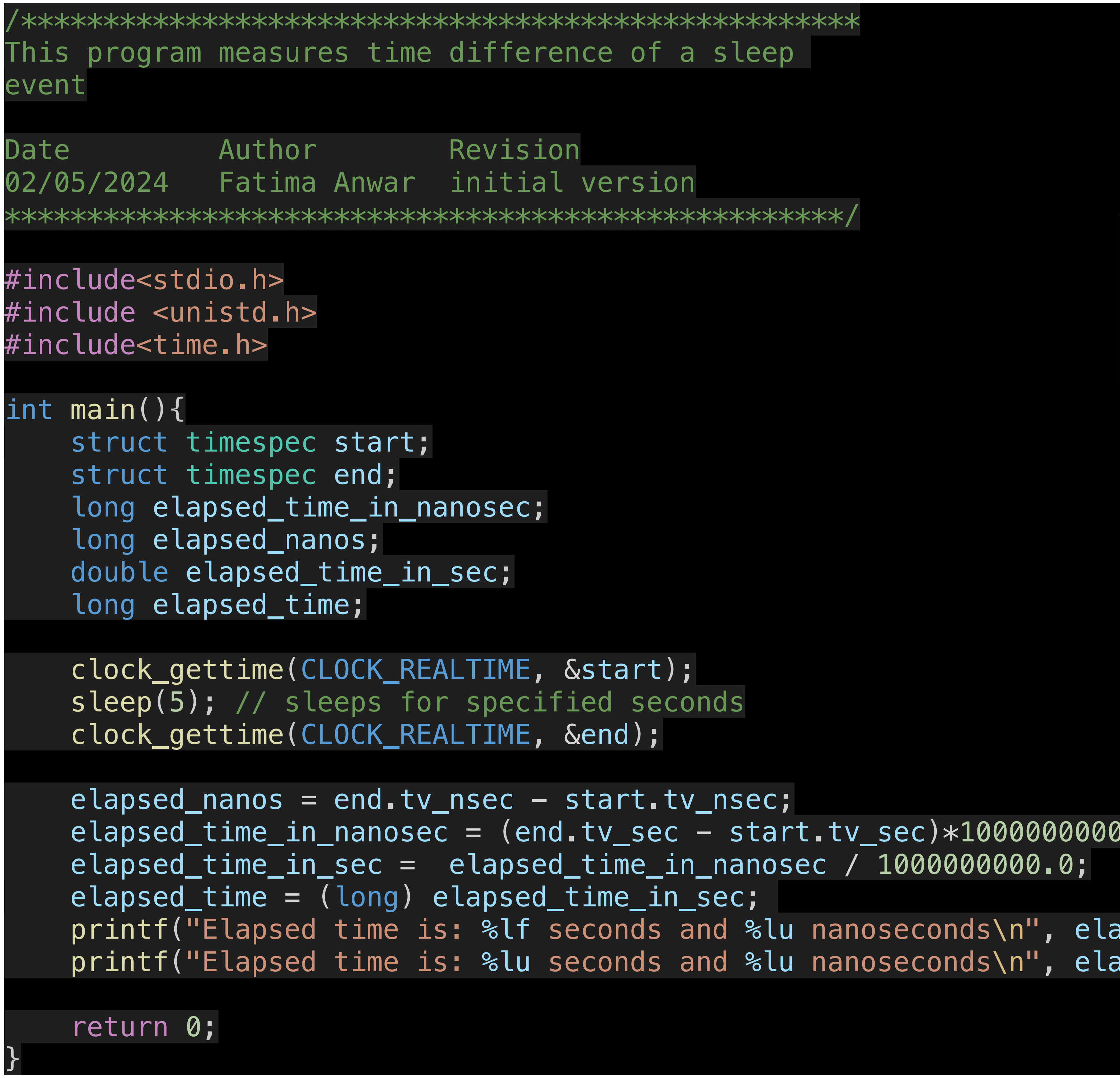
Note: this declares a new **data type**, not a variable

```
struct colour white;

struct colour black;
```

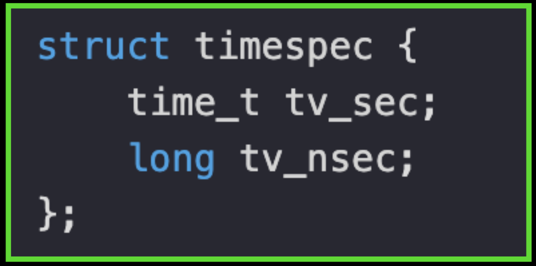
To create a variable of type colour, we declare it with struct type

```
white.r = 255;

white.b = 120;
```

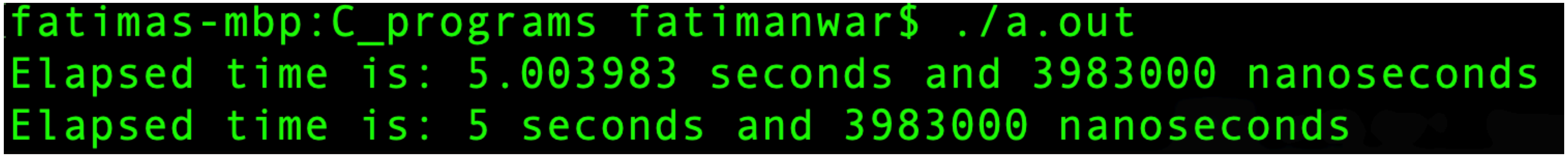
We access the members by writing the name of the structure variable followed by a dot and the name of the member



```c
/*****************************************************
This program measures time difference of a sleep 
event

Date          Author          Revision
02/05/2024    Fatima Anwar    initial version
*****************************************************/

#include<stdio.h>
#include <unistd.h>
#include<time.h>

int main(){
    struct timespec start;
    struct timespec end;
    long elapsed_time_in_nanosec;
    long elapsed_nanos;
    double elapsed_time_in_sec;
    long elapsed_time;

    clock_gettime(CLOCK_REALTIME, &start);
    sleep(5); // sleeps for specified seconds
    clock_gettime(CLOCK_REALTIME, &end);

    elapsed_nanos = end.tv_nsec - start.tv_nsec;
    elapsed_time_in_nanosec = (end.tv_sec - start.tv_sec)*1000000000 + elapsed_nanos;
    elapsed_time_in_sec =  elapsed_time_in_nanosec / 1000000000.0;
    elapsed_time = (long) elapsed_time_in_sec;
    printf("Elapsed time is: %lf seconds and %lu nanoseconds\n", elapsed_time_in_sec, elapsed_nanos);
    printf("Elapsed time is: %lu seconds and %lu nanoseconds\n", elapsed_time, elapsed_nanos);

    return 0;
}
```

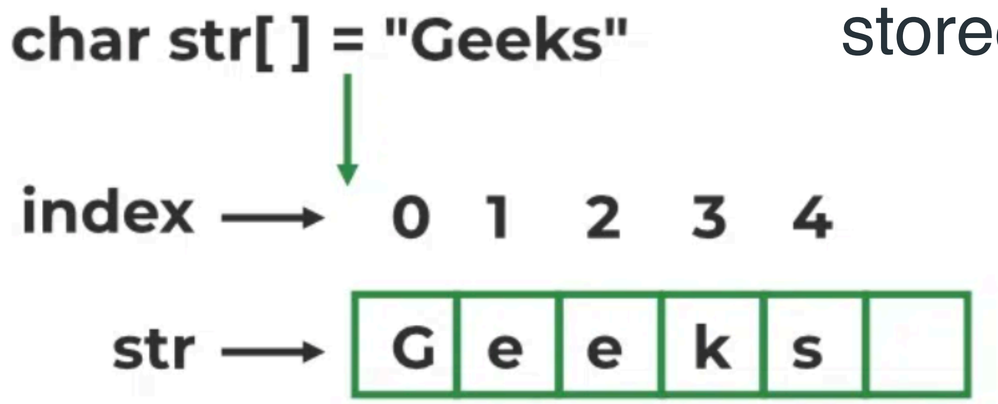


```c
struct timespec {
    time_t tv_sec;
    long tv_nsec;
};
```



```
fatimas-mbp:C_programs fatimanwar$ ./a.out
Elapsed time is: 5.003983 seconds and 3983000 nanoseconds
Elapsed time is: 5 seconds and 3983000 nanoseconds
```

String



stored as an array of characters

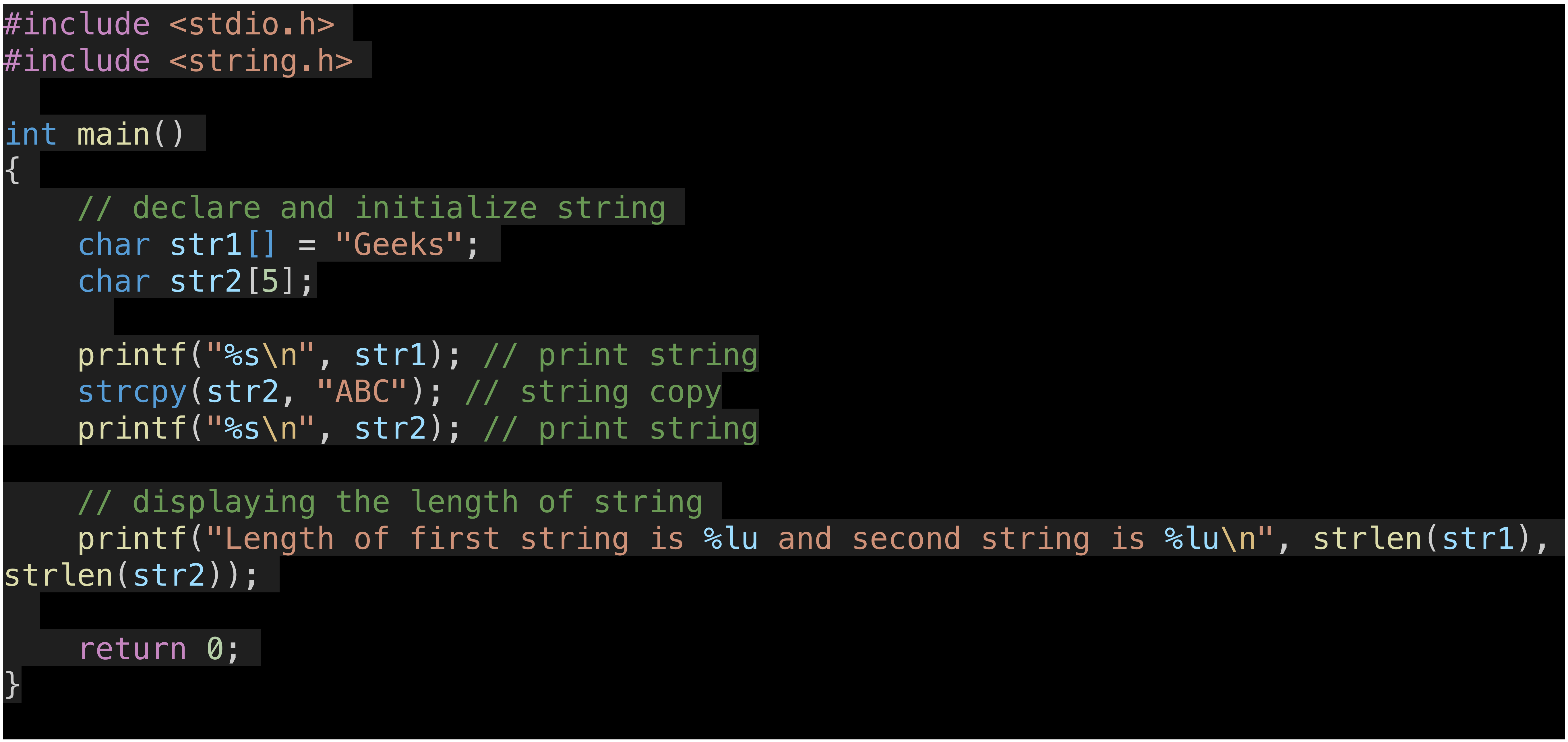
- strcat(destination, source); // concatenate source to destination
- strcmp(string1, string2);

  // returns 0 in the case of equality

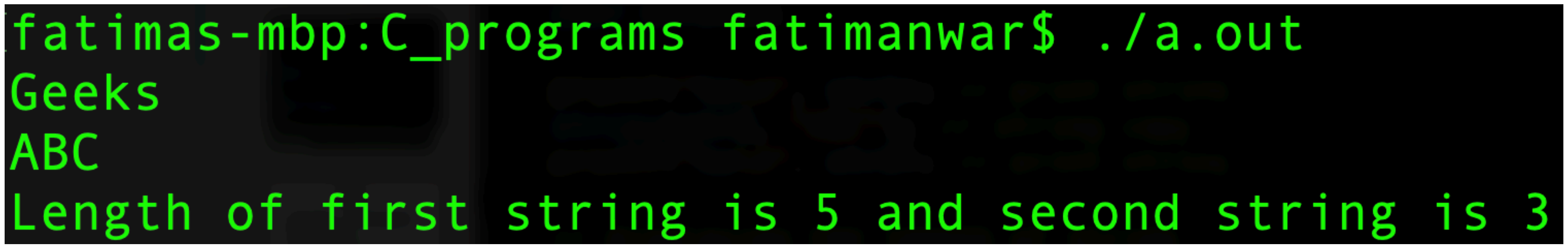
  // returns <0 if string1 < string2

  // returns >0 if string1 > string2

- strlen(string); // returns the length of the string

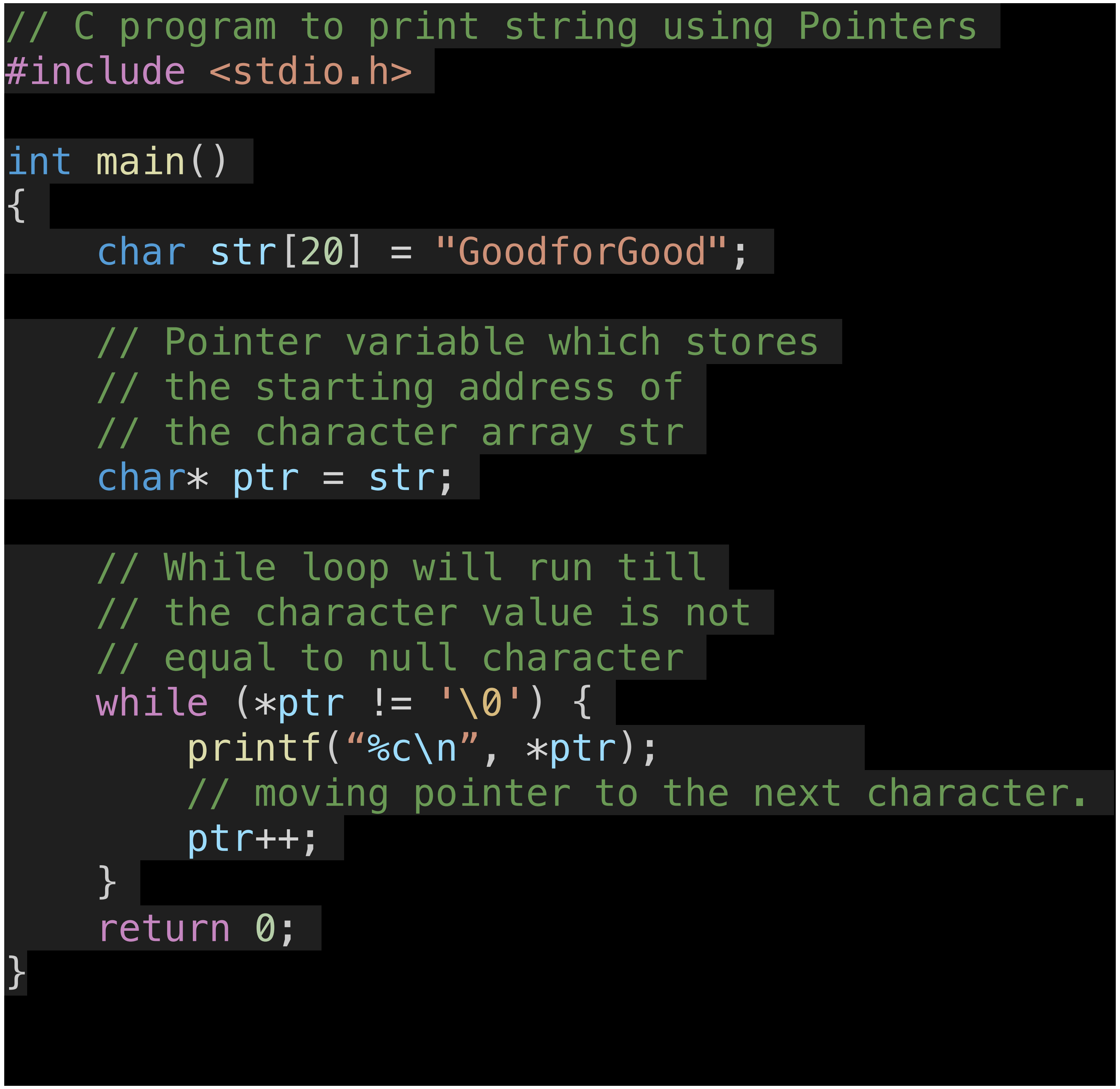


```c
#include <stdio.h>
#include <string.h>

int main()
{
    // declare and initialize string
    char str1[] = "Geeks";
    char str2[5];

    printf("%s\n", str1); // print string
    strcpy(str2, "ABC"); // string copy
    printf("%s\n", str2); // print string

    // displaying the length of string
    printf("Length of first string is %lu and second string is %lu\n", strlen(str1),
strlen(str2));

    return 0;
}
```

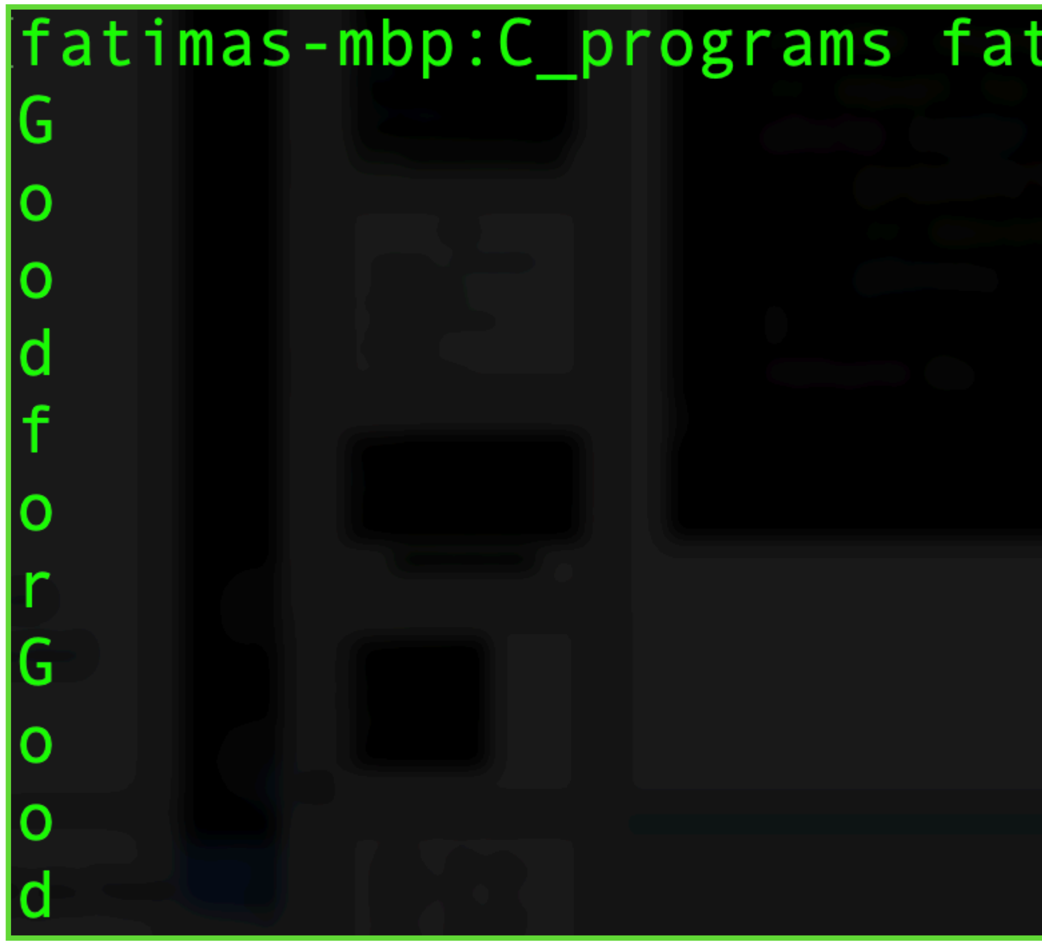


```
fatimas-mbp:C_programs fatimanwar$ ./a.out
Geeks
ABC
Length of first string is 5 and second string is 3
```

Use pointer to print string

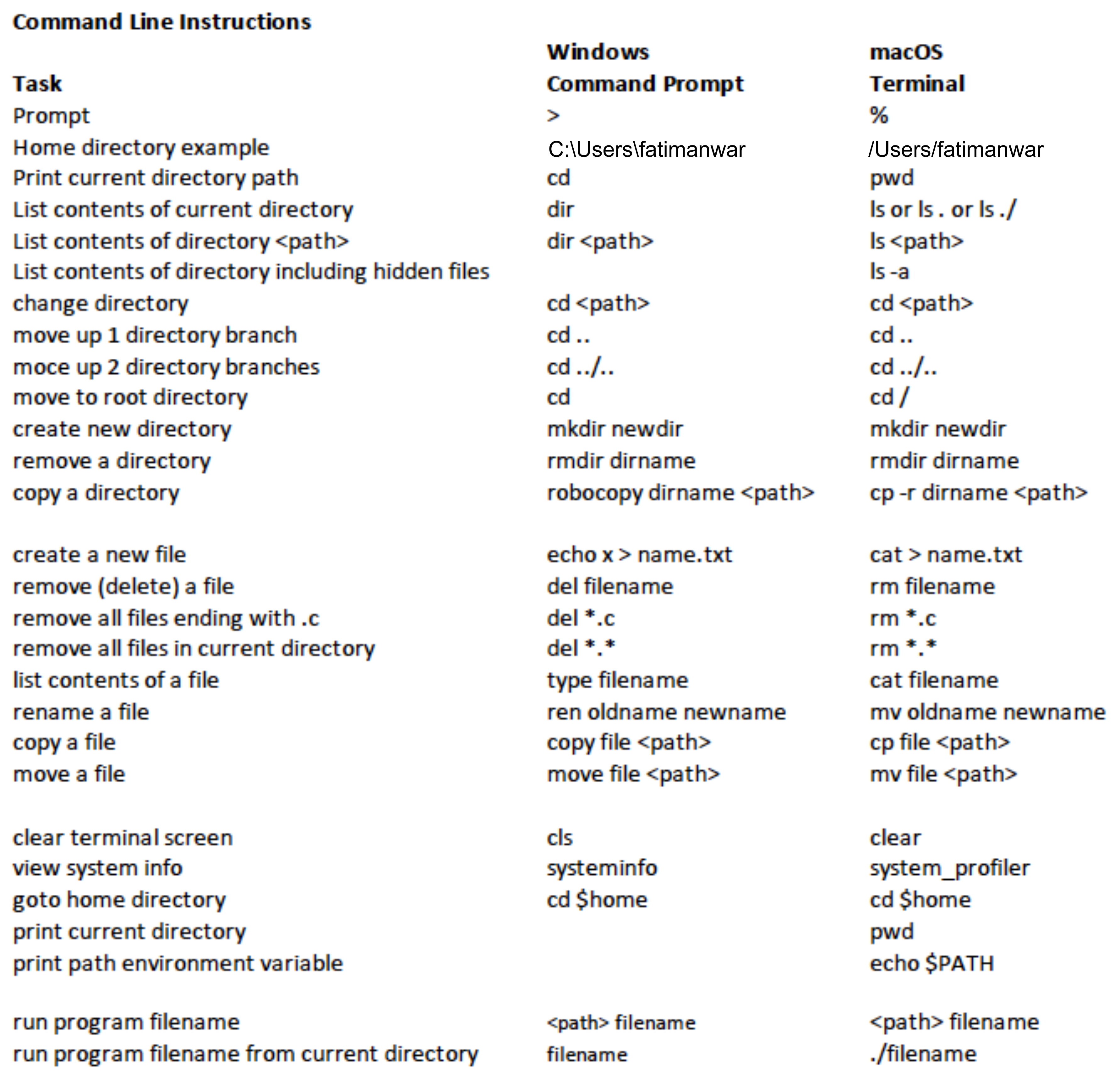


```c
// C program to print string using Pointers
#include <stdio.h>

int main()
{
    char str[20] = "GoodforGood";

    // Pointer variable which stores
    // the starting address of
    // the character array str
    char* ptr = str;

    // While loop will run till
    // the character value is not
    // equal to null character
    while (*ptr != '\0') {
        printf(“%c\n”, *ptr);
        // moving pointer to the next character.
        ptr++;
    }
    return 0;
}
```



```
fatimas-mbp:C_programs fat
G
o
o
d
f
o
r
G
o
o
d
```

Most commonly used Linux instructions for command line



Command Line Instructions

Task	**Windows Command Prompt**	**macOS Terminal**
Prompt	>	%
Home directory example	C:\Users\fatimanwar	/Users/fatimanwar
Print current directory path	cd	pwd
List contents of current directory	dir	ls or ls . or ls ./
List contents of directory <path>	dir <path>	ls <path>
List contents of directory including hidden files		ls -a
change directory	cd <path>	cd <path>
move up 1 directory branch	cd ..	cd ..
moce up 2 directory branches	cd ../..	cd ../..
move to root directory	cd	cd /
create new directory	mkdir newdir	mkdir newdir
remove a directory	rmdir dirname	rmdir dirname
copy a directory	robocopy dirname <path>	cp -r dirname <path>
create a new file	echo x > name.txt	cat > name.txt
remove (delete) a file	del filename	rm filename
remove all files ending with .c	del *.c	rm *.c
remove all files in current directory	del *.*	rm *.*
list contents of a file	type filename	cat filename
rename a file	ren oldname newname	mv oldname newname
copy a file	copy file <path>	cp file <path>
move a file	move file <path>	mv file <path>
clear terminal screen	cls	clear
view system info	systeminfo	system_profiler
goto home directory	cd $home	cd $home
print current directory		pwd
print path environment variable		echo $PATH
run program filename	<path> filename	<path> filename
run program filename from current directory	filename	./filename

C Language Keywords

`auto`	`break`	`case`	`char`	`const`
`continue`	`default`	`do`	`double`	`else`
`enum`	`extern`	`float`	`for`	`goto`
`if`	`int`	`long`	`register`	`return`
`short`	`signed`	`sizeof`	`static`	`struct`
`switch`	`typedef`	`union`	`unsigned`	`void`
`volatile`	`while`			