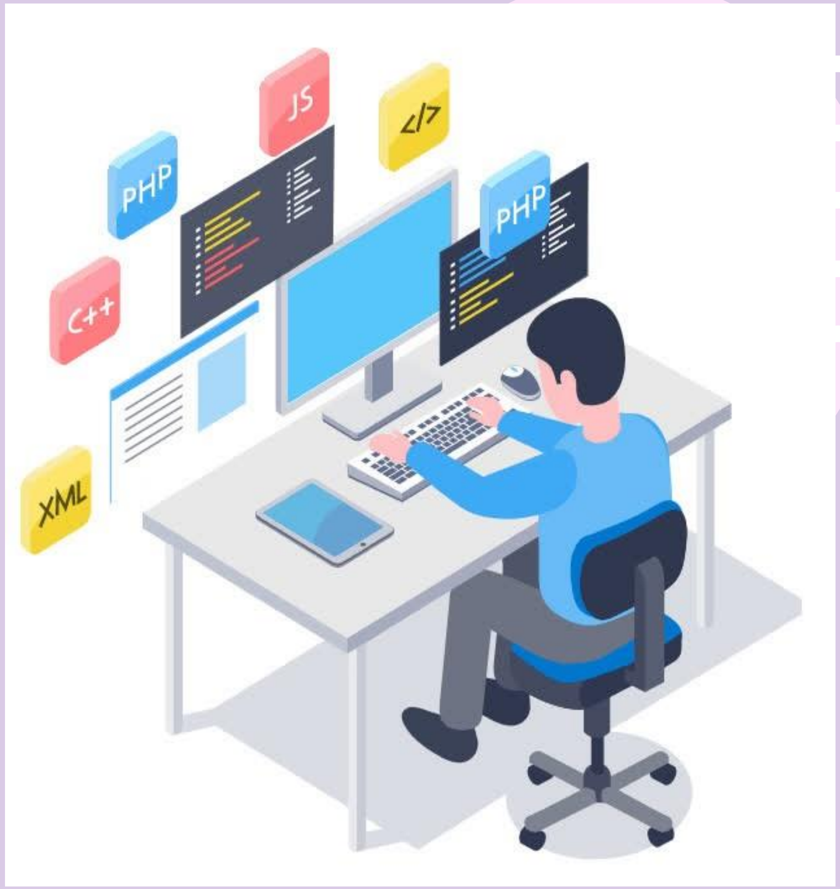


Uva Wellassa University

Programming Techniques

ICT 131-3

Practical 7 - C Programming VII

Outline -

What we are going to learn on today ?

- C Math Functions
- C Two-Dimensional Arrays
- C Pointers

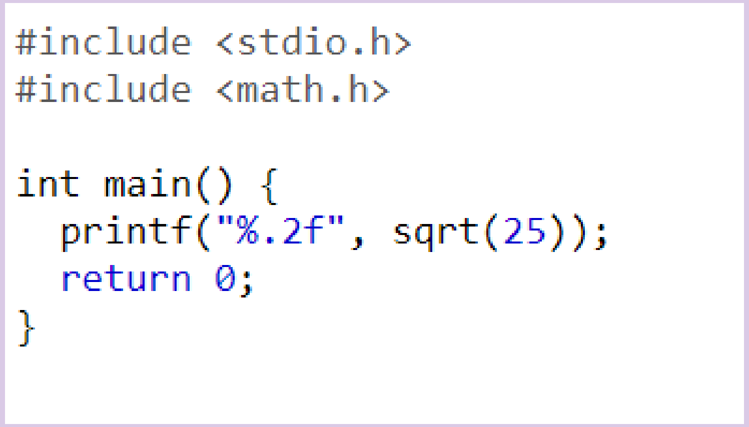
C Math Functions

- There is also a list of math functions available, that allows you to perform mathematical tasks on numbers.
- To use them, you must include the math.h

#include <math.h>

Example 1

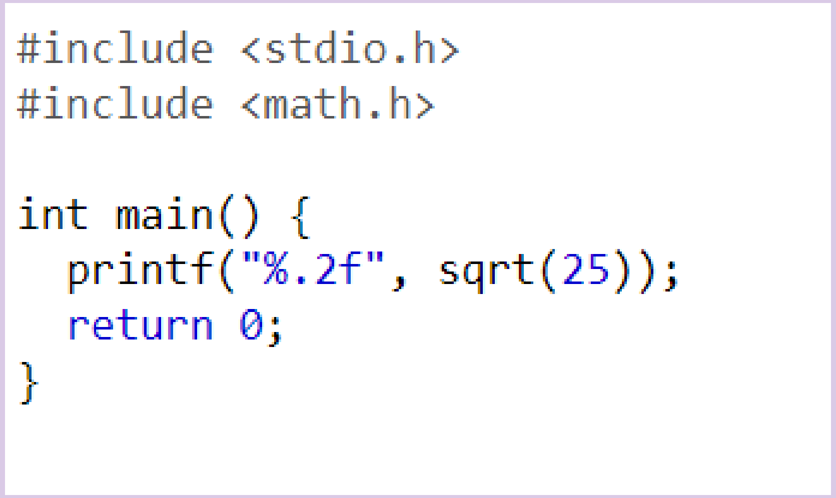
To find the square root of 25, use the sqrt() function.



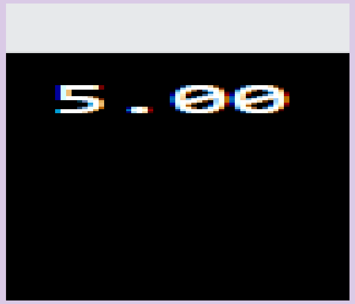
```
#include <stdio.h>
#include <math.h>

int main() {
  printf("%.2f", sqrt(25));
  return 0;
}
```

Example 1

To find the square root of 25, use the sqrt() function.



```
#include <stdio.h>
#include <math.h>

int main() {
  printf("%.2f", sqrt(25));
  return 0;
}
```



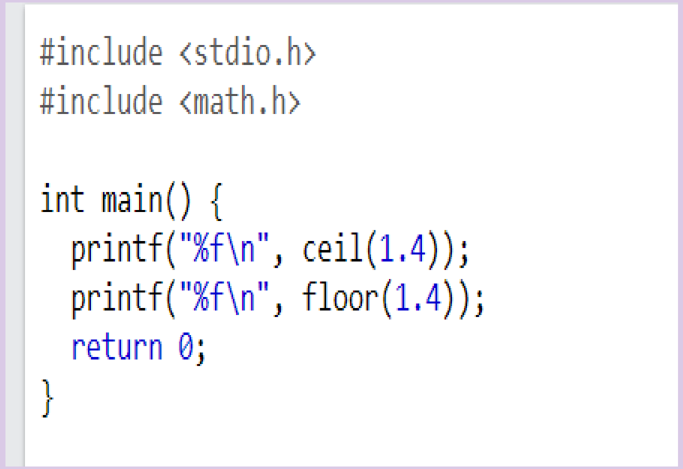
```
5.00
```

Example 2

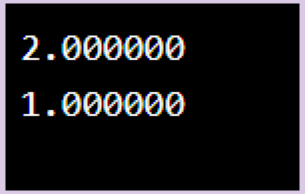
The ceil() function rounds a number upwards to its nearest integer, and the floor() method rounds a number downwards to its nearest integer, and returns the result

Example 2

The ceil() function rounds a number upwards to its nearest integer, and the floor() method rounds a number downwards to its nearest integer, and returns the result



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

int main() {
  printf("%f\n", ceil(1.4));
  printf("%f\n", floor(1.4));
  return 0;
}
```



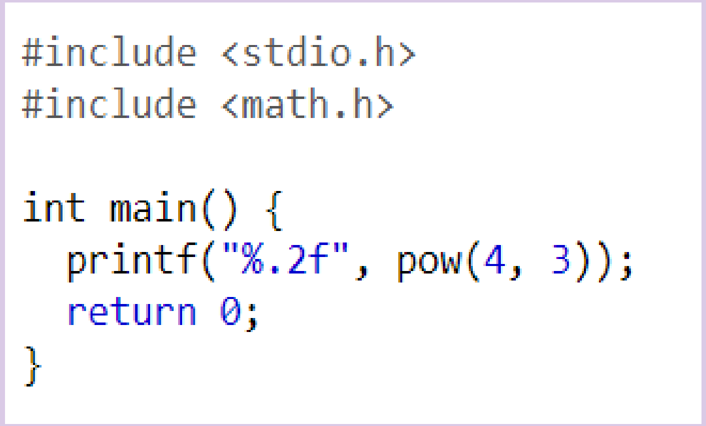
```
2.000000
1.000000
```

Example 3

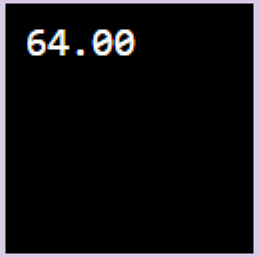
The pow() function returns the value of *x* to the power of *y* (x^y)

Example 3

The pow() function returns the value of *x* to the power of *y* (x^y)



```
#include <stdio.h>
#include <math.h>

int main() {
  printf("%.2f", pow(4, 3));
  return 0;
}
```

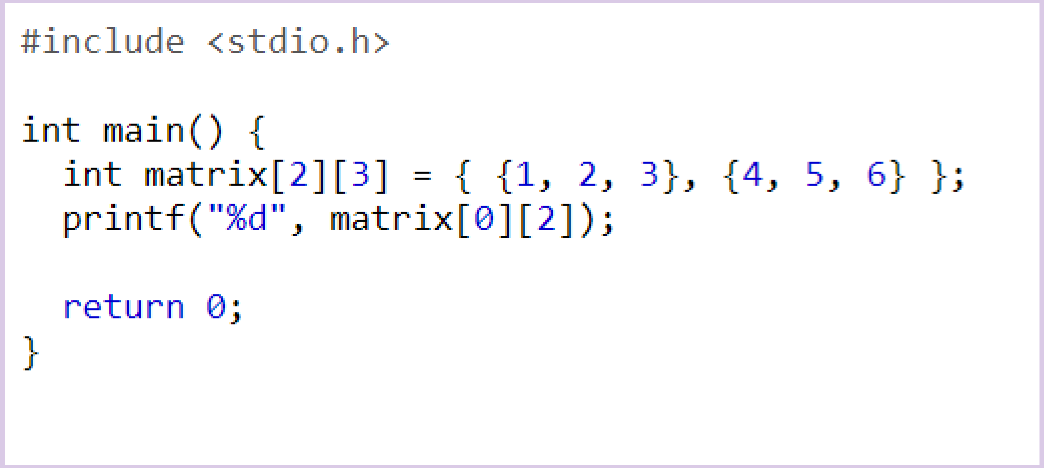


```
64.00
```

C Two-Dimensional Arrays

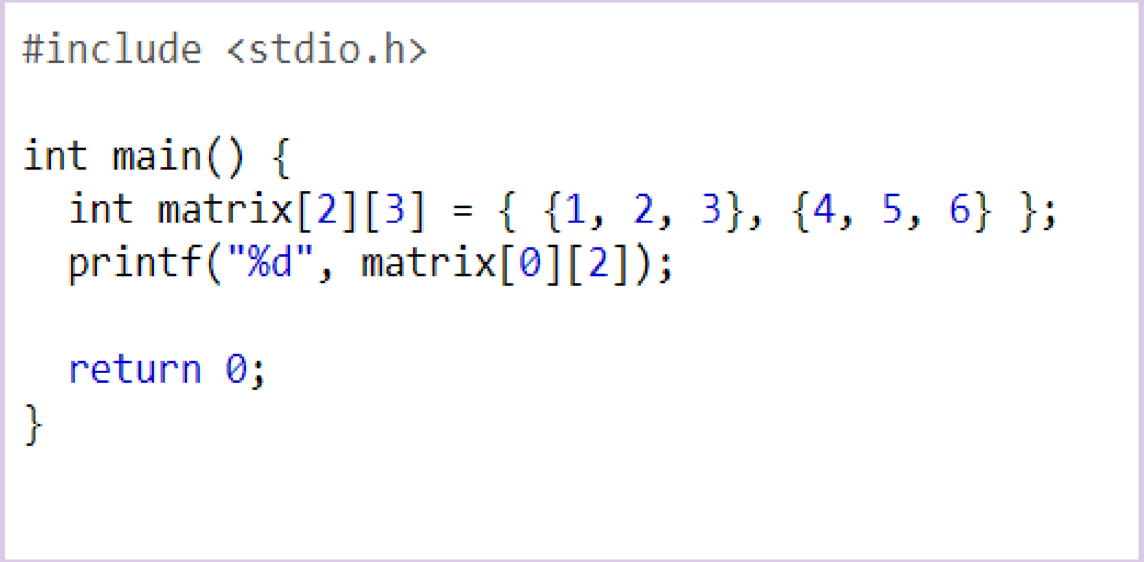
- A 2D array is also known as a matrix (a table of rows and columns).

Example 4

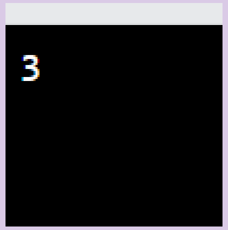


```c
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };
  printf("%d", matrix[0][2]);

  return 0;
}
```

Example 4



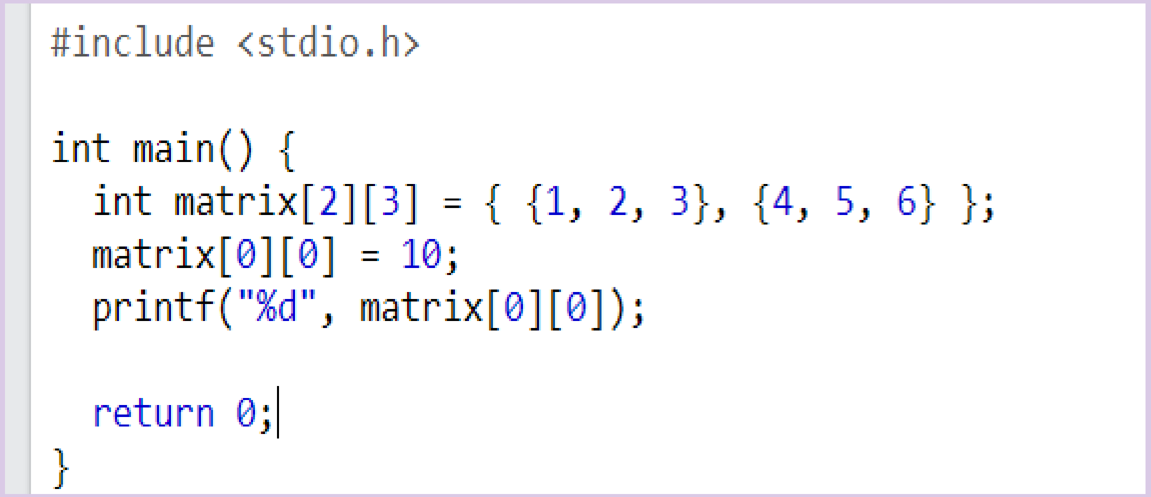
```c
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };
  printf("%d", matrix[0][2]);

  return 0;
}
```



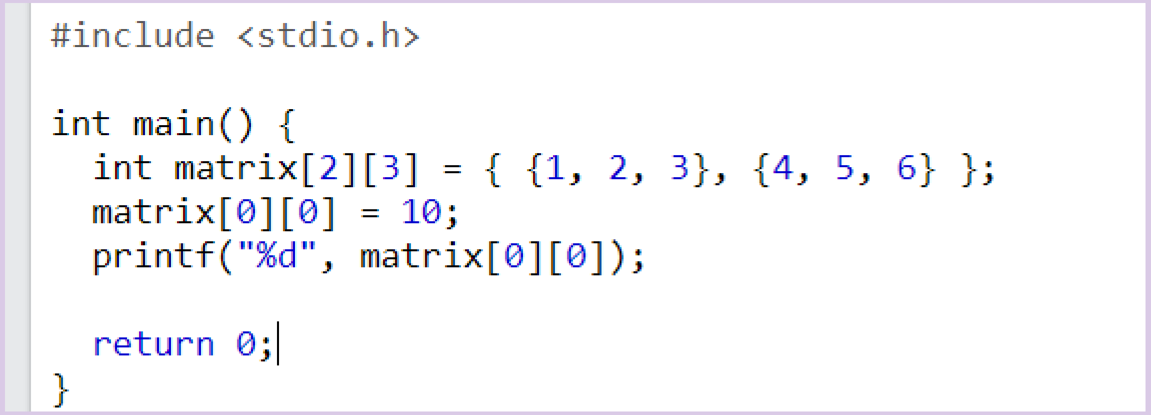
```
3
```

Change Elements in a 2D Array

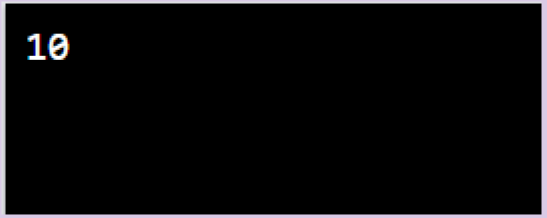
Example 5


```
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };
  matrix[0][0] = 10;
  printf("%d", matrix[0][0]);

  return 0;
}
```

Example 5



```c
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };
  matrix[0][0] = 10;
  printf("%d", matrix[0][0]);

  return 0;
}
```

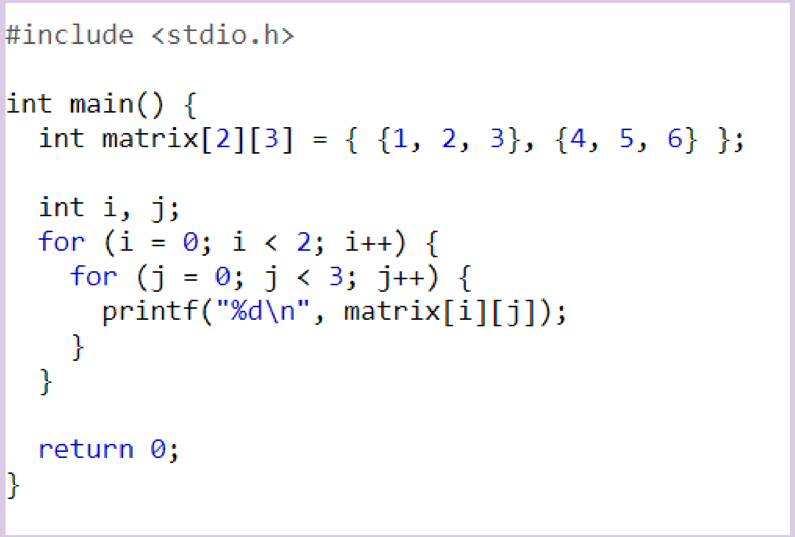


```
10
```

Loop Through a 2D Array

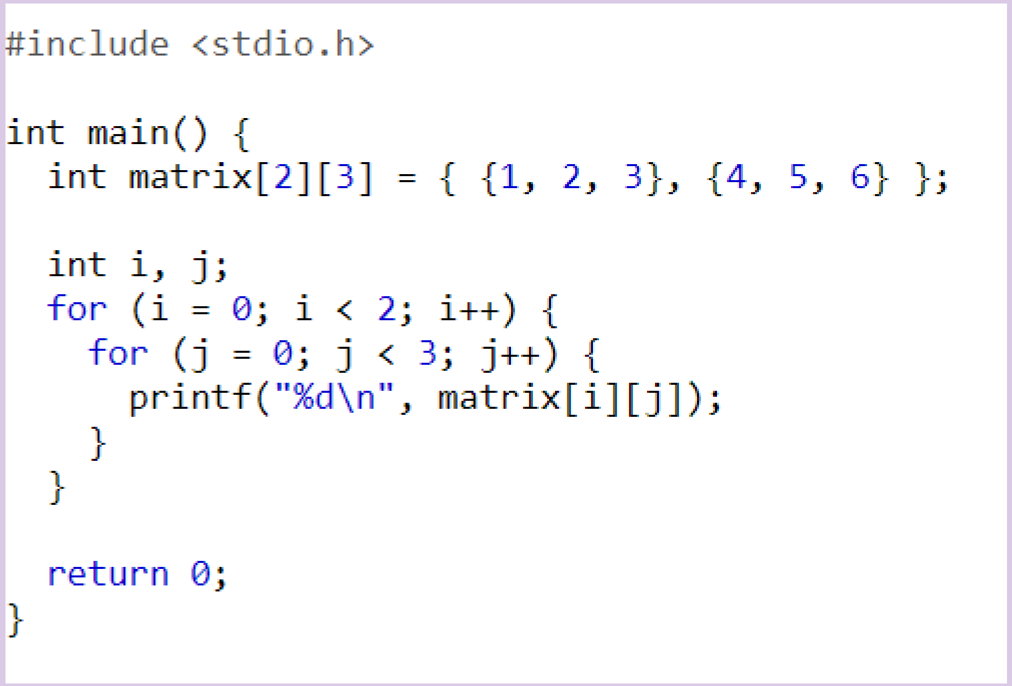
To loop through a multi-dimensional array, you need one loop for each of the array's dimensions.

Example 6

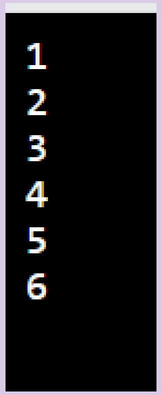


```c
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };

  int i, j;
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
      printf("%d\n", matrix[i][j]);
    }
  }

  return 0;
}
```

Example 6



```c
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };

  int i, j;
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
      printf("%d\n", matrix[i][j]);
    }
  }

  return 0;
}
```



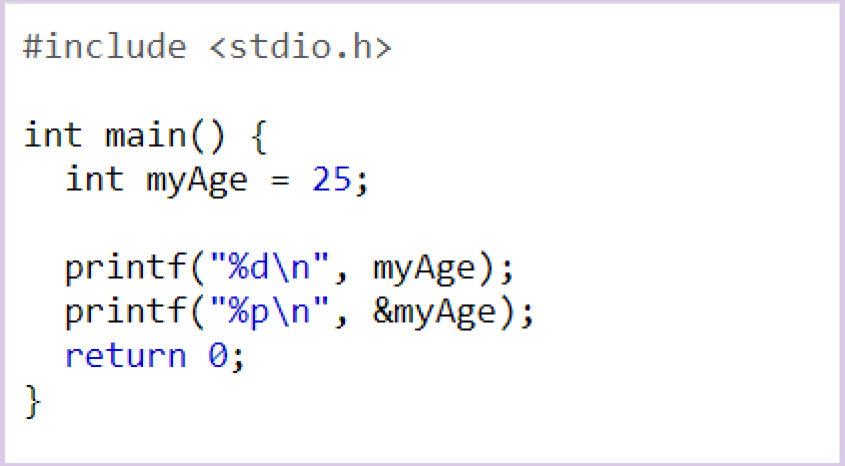
```
1
2
3
4
5
6
```

C Pointers

Creating Pointers

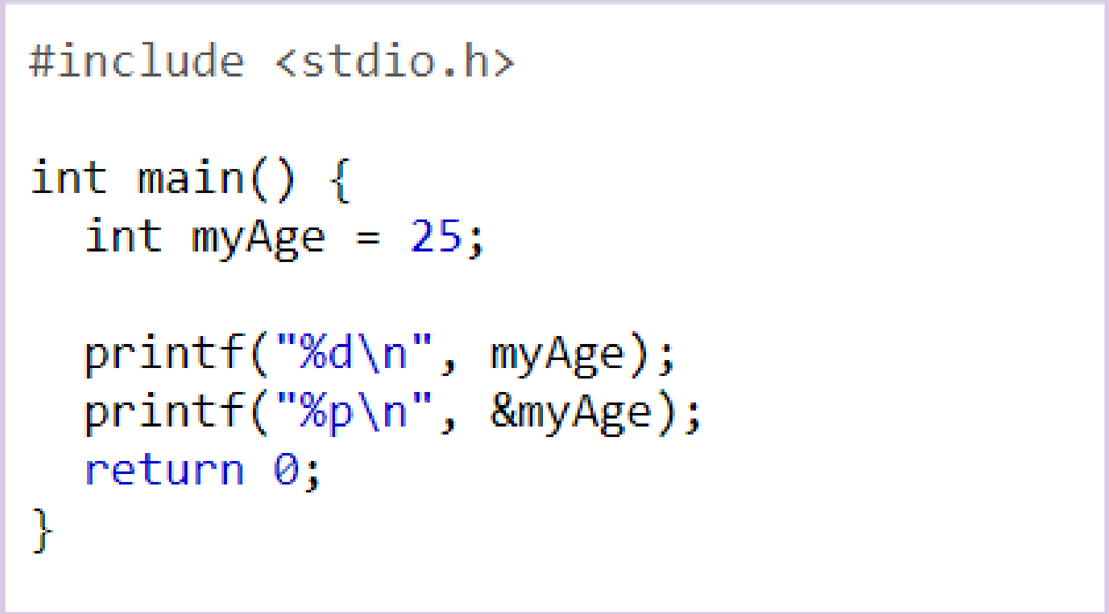
We can get the **memory address** of a variable with the reference operator &

Example 7

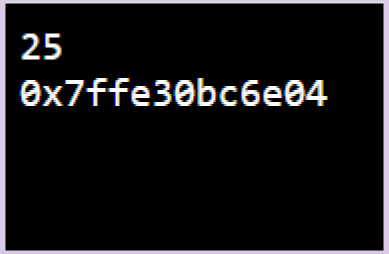


```
#include <stdio.h>

int main() {
  int myAge = 25;

  printf("%d\n", myAge);
  printf("%p\n", &myAge);
  return 0;
}
```

Example 7

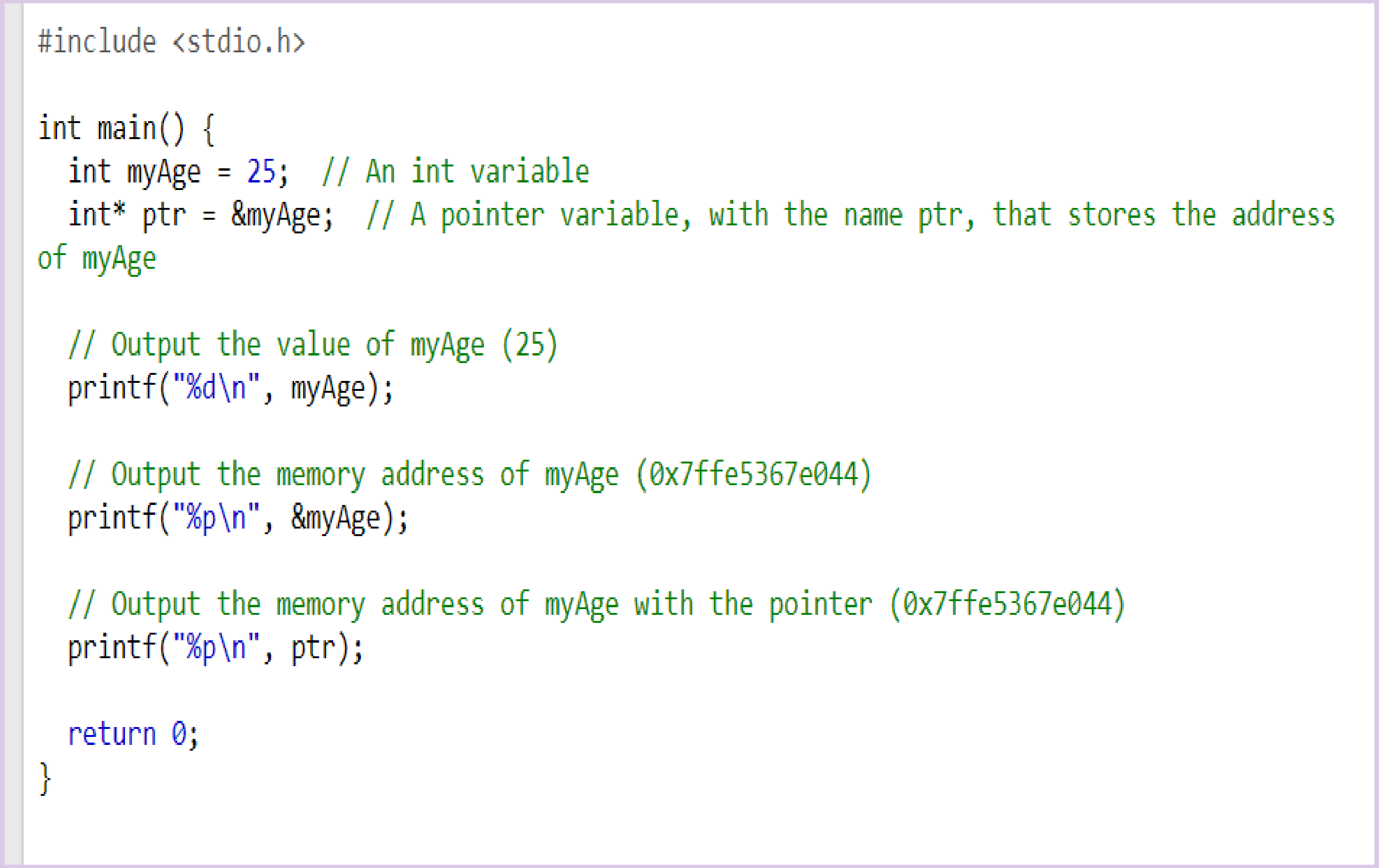


```c
#include <stdio.h>

int main() {
  int myAge = 25;

  printf("%d\n", myAge);
  printf("%p\n", &myAge);
  return 0;
}
```



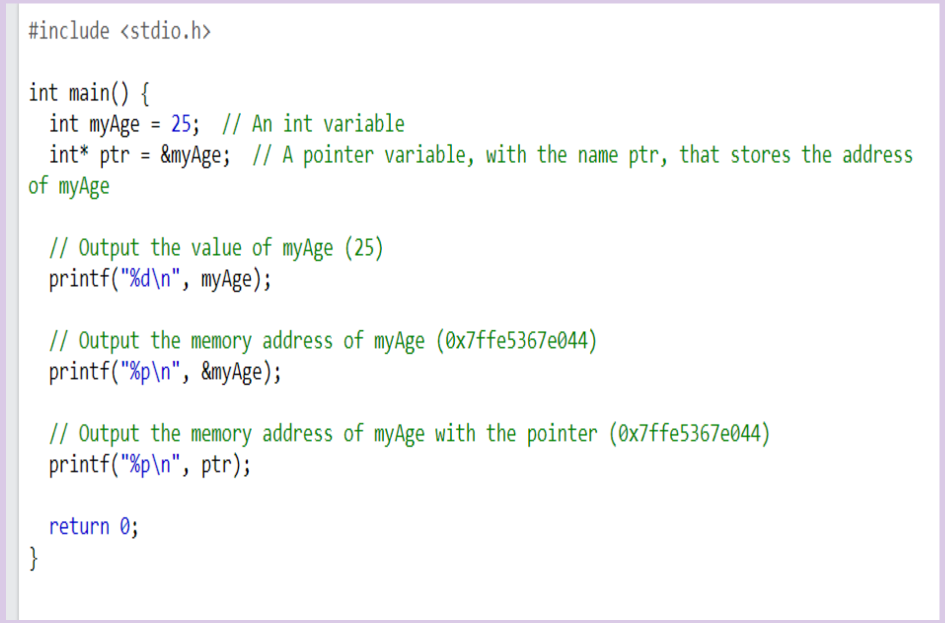
```
25
0x7ffe30bc6e04
```

Example 8

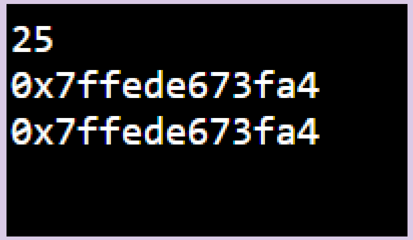


```c
#include <stdio.h>

int main() {
  int myAge = 25;  // An int variable
  int* ptr = &myAge;  // A pointer variable, with the name ptr, that stores the address
of myAge

  // Output the value of myAge (25)
  printf("%d\n", myAge);

  // Output the memory address of myAge (0x7ffe5367e044)
  printf("%p\n", &myAge);

  // Output the memory address of myAge with the pointer (0x7ffe5367e044)
  printf("%p\n", ptr);

  return 0;
}
```

Example 8



```c
#include <stdio.h>

int main() {
  int myAge = 25;  // An int variable
  int* ptr = &myAge;  // A pointer variable, with the name ptr, that stores the address
of myAge

  // Output the value of myAge (25)
  printf("%d\n", myAge);

  // Output the memory address of myAge (0x7ffe5367e044)
  printf("%p\n", &myAge);

  // Output the memory address of myAge with the pointer (0x7ffe5367e044)
  printf("%p\n", ptr);

  return 0;
}
```

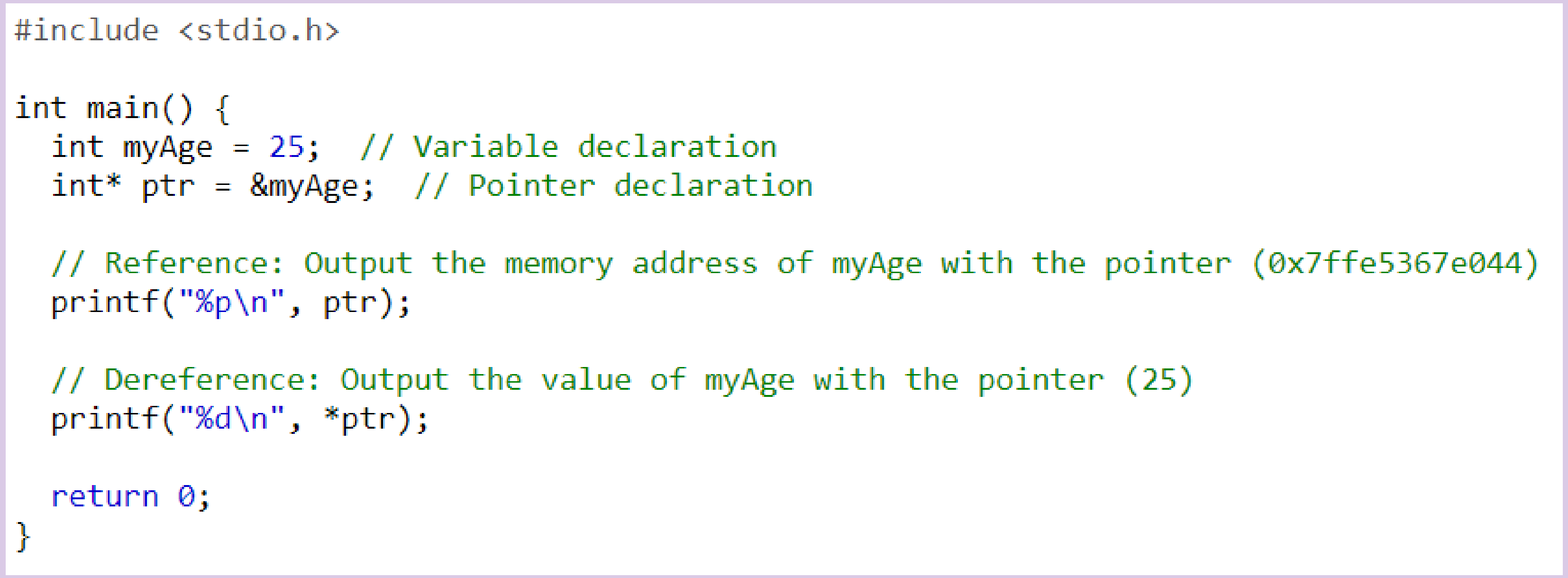


```
25
0x7ffede673fa4
0x7ffede673fa4
```

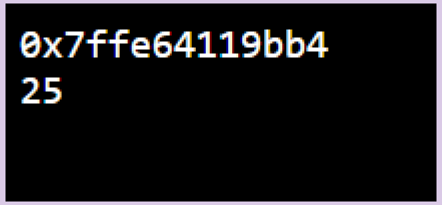
Dereference

- You can also get the value of the variable the pointer points to, by using the * operator (the **dereference** operator)

Example 9

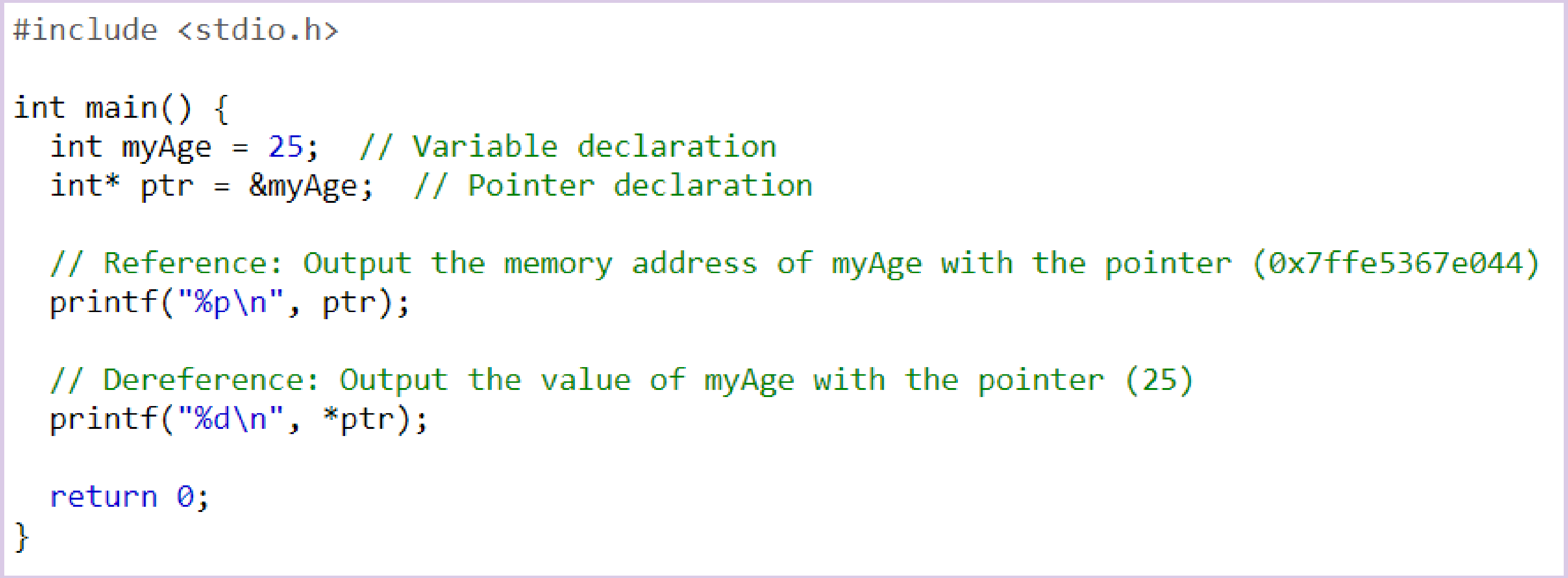


```c
#include <stdio.h>

int main() {
  int myAge = 25;  // Variable declaration
  int* ptr = &myAge;  // Pointer declaration

  // Reference: Output the memory address of myAge with the pointer (0x7ffe5367e044)
  printf("%p\n", ptr);

  // Dereference: Output the value of myAge with the pointer (25)
  printf("%d\n", *ptr);

  return 0;
}
```

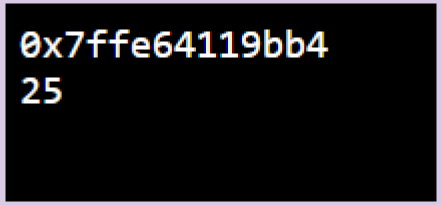


```
0x7ffe64119bb4
25
```

Example 9



```c
#include <stdio.h>

int main() {
  int myAge = 25;  // Variable declaration
  int* ptr = &myAge;  // Pointer declaration

  // Reference: Output the memory address of myAge with the pointer (0x7ffe5367e044)
  printf("%p\n", ptr);

  // Dereference: Output the value of myAge with the pointer (25)
  printf("%d\n", *ptr);

  return 0;
}
```

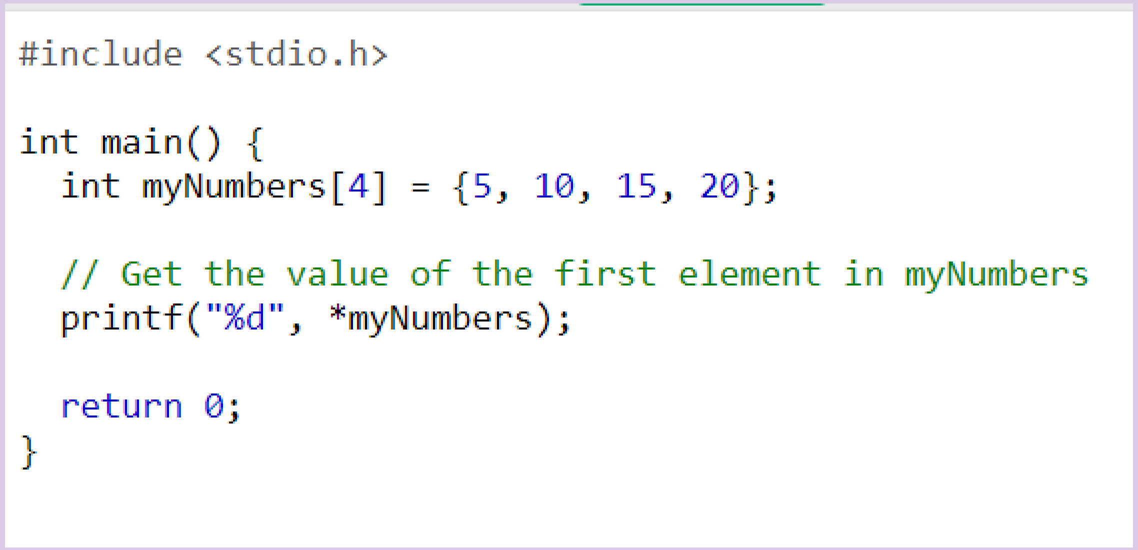


```
0x7ffe64119bb4
25
```

C Pointers and Arrays

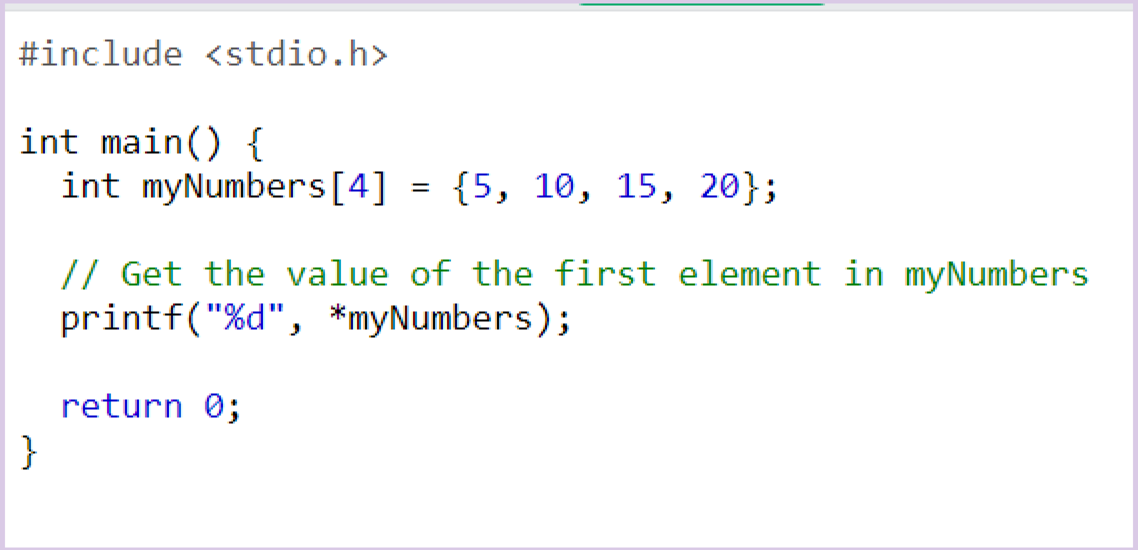
- You can also use pointers to access arrays.

Example 10

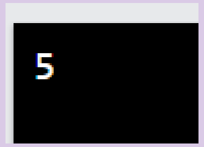


```
#include <stdio.h>

int main() {
  int myNumbers[4] = {5, 10, 15, 20};

  // Get the value of the first element in myNumbers
  printf("%d", *myNumbers);

  return 0;
}
```

Example 10

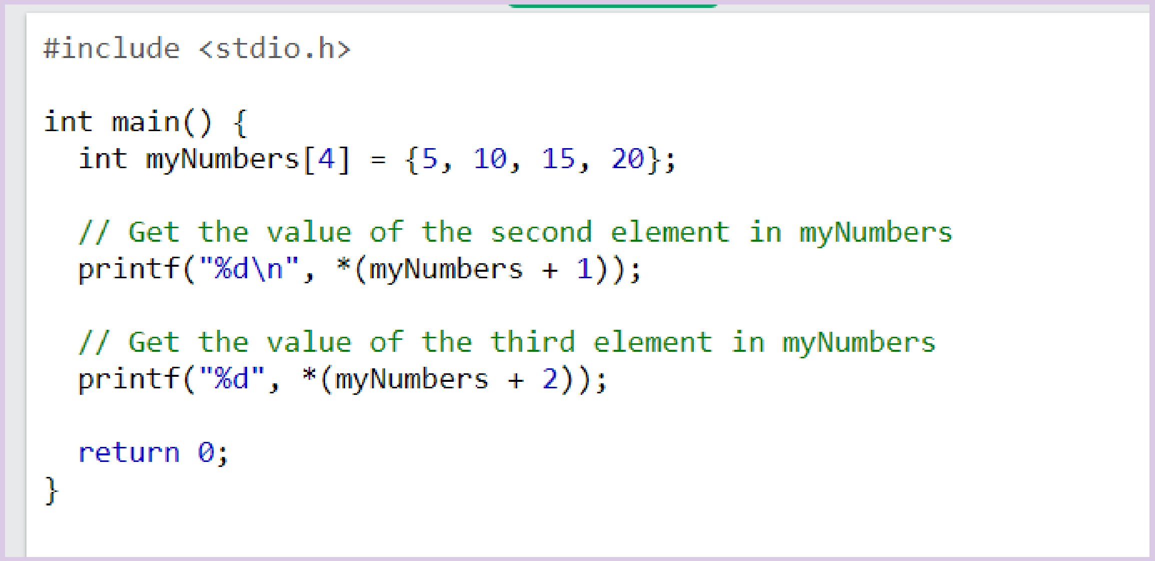


```c
#include <stdio.h>

int main() {
  int myNumbers[4] = {5, 10, 15, 20};

  // Get the value of the first element in myNumbers
  printf("%d", *myNumbers);

  return 0;
}
```



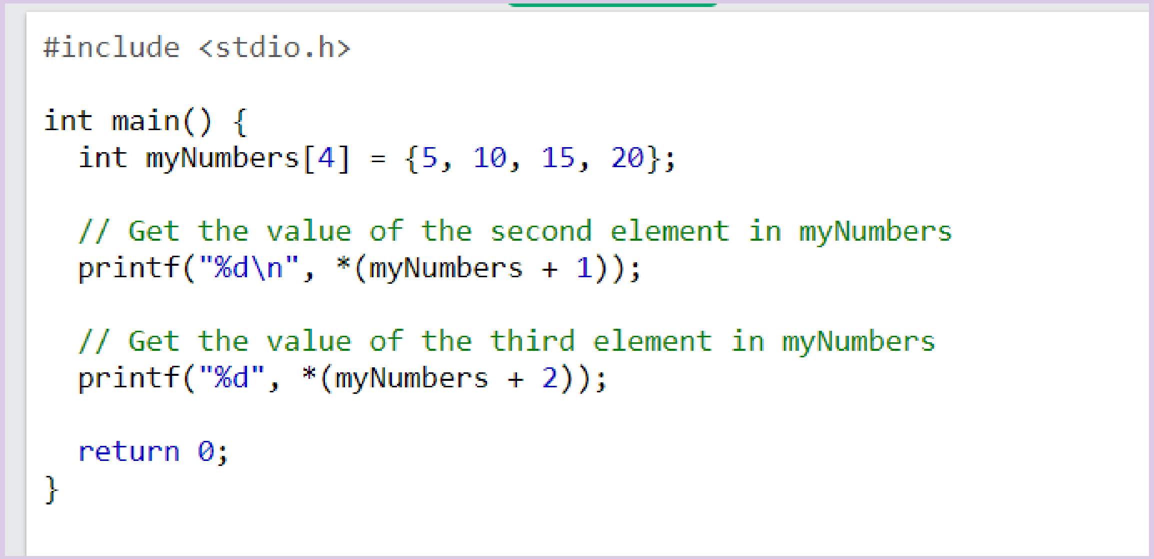
```
5
```

Example 11

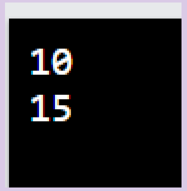


```c
#include <stdio.h>

int main() {
  int myNumbers[4] = {5, 10, 15, 20};

  // Get the value of the second element in myNumbers
  printf("%d\n", *(myNumbers + 1));

  // Get the value of the third element in myNumbers
  printf("%d", *(myNumbers + 2));

  return 0;
}
```

Example 11



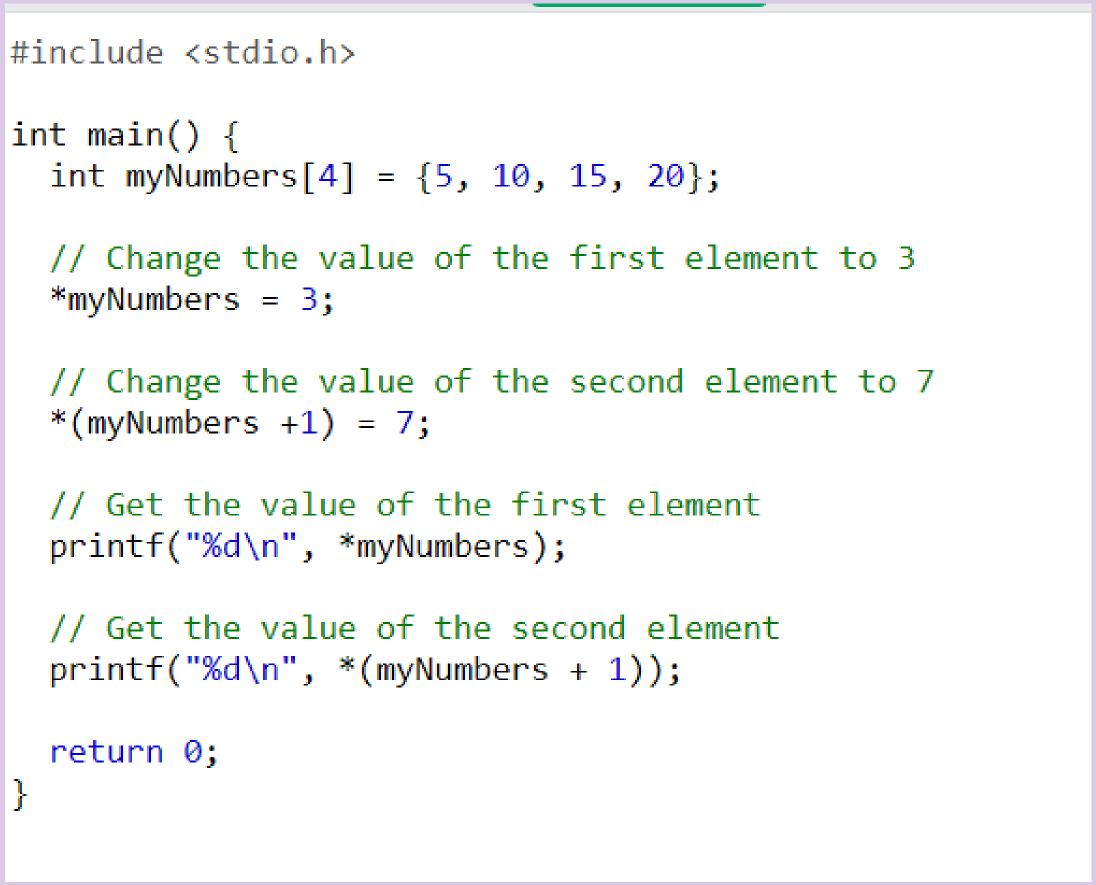
```
#include <stdio.h>

int main() {
  int myNumbers[4] = {5, 10, 15, 20};

  // Get the value of the second element in myNumbers
  printf("%d\n", *(myNumbers + 1));

  // Get the value of the third element in myNumbers
  printf("%d", *(myNumbers + 2));

  return 0;
}
```



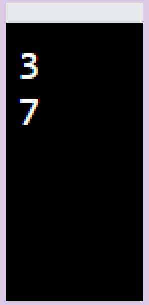
```
10
15
```

Change the value of array elements with pointers

Example 12

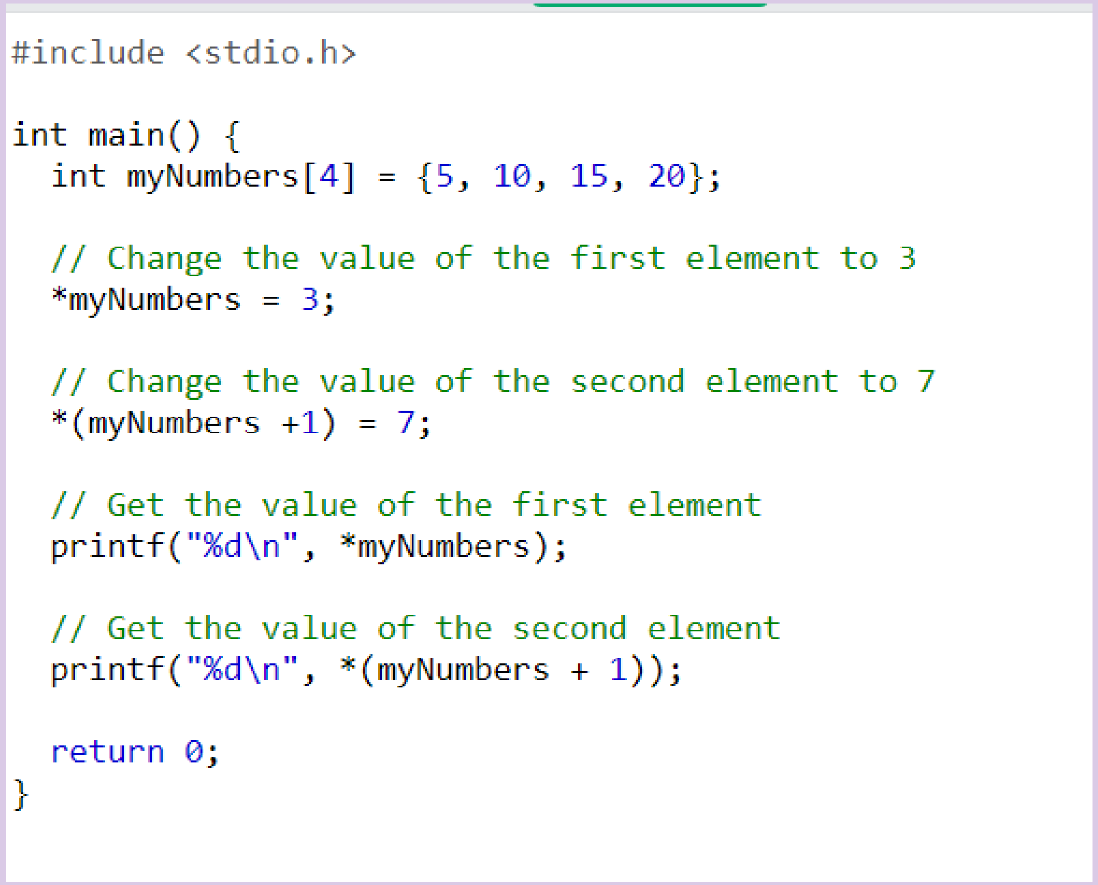


```c
#include <stdio.h>

int main() {
  int myNumbers[4] = {5, 10, 15, 20};

  // Change the value of the first element to 3
  *myNumbers = 3;

  // Change the value of the second element to 7
  *(myNumbers +1) = 7;

  // Get the value of the first element
  printf("%d\n", *myNumbers);

  // Get the value of the second element
  printf("%d\n", *(myNumbers + 1));

  return 0;
}
```

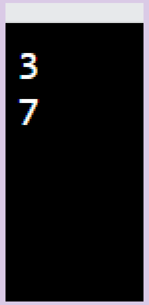


```
3
7
```

Example 12



```c
#include <stdio.h>

int main() {
  int myNumbers[4] = {5, 10, 15, 20};

  // Change the value of the first element to 3
  *myNumbers = 3;

  // Change the value of the second element to 7
  *(myNumbers +1) = 7;

  // Get the value of the first element
  printf("%d\n", *myNumbers);

  // Get the value of the second element
  printf("%d\n", *(myNumbers + 1));

  return 0;
}
```



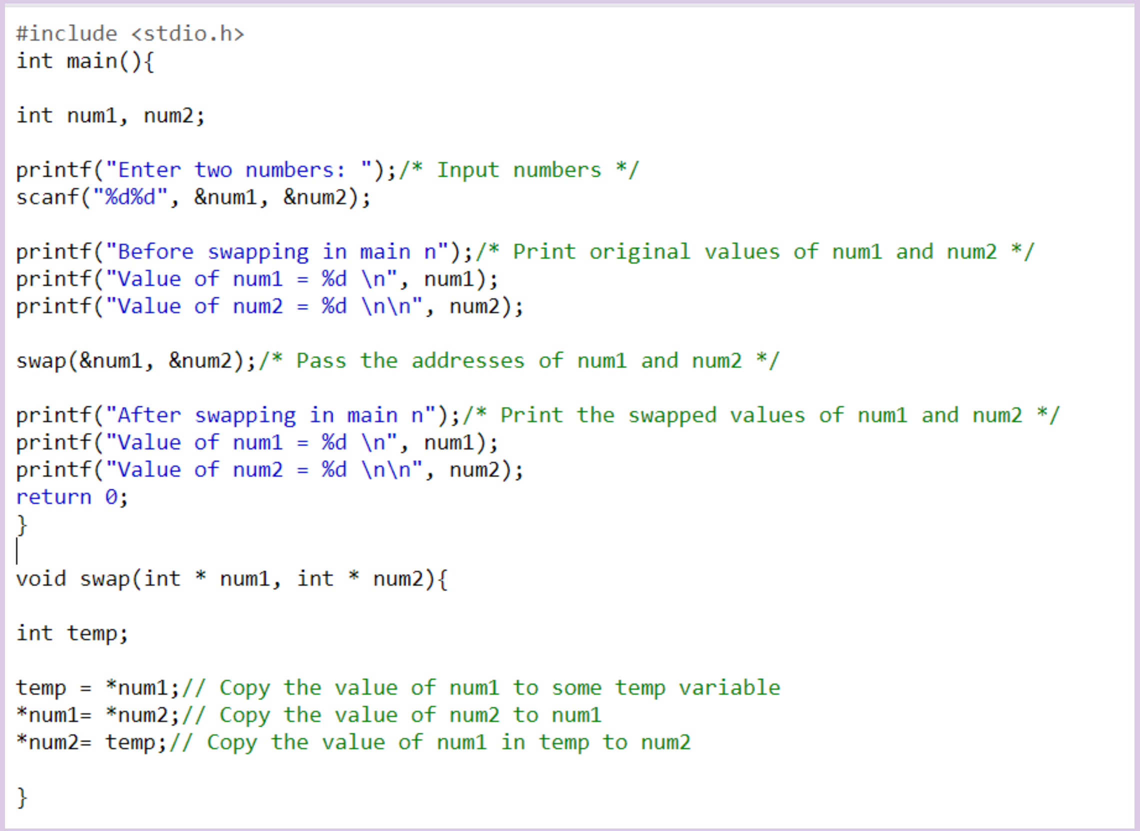
```
3
7
```

Example 13

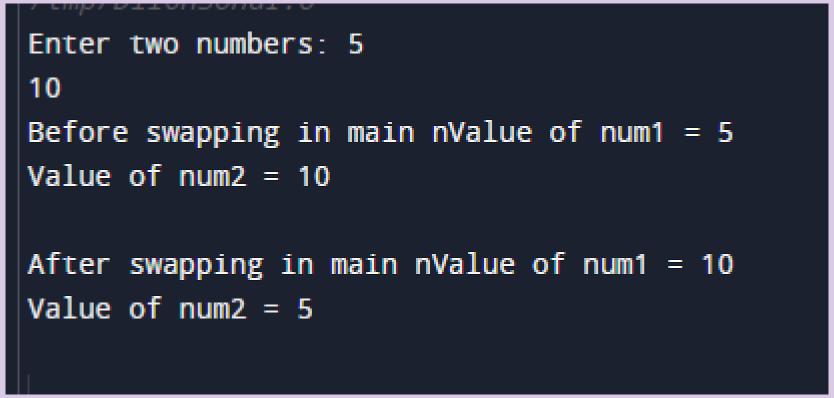
Swap two numbers using pointers

Example 12

Swap two numbers using pointers



```c
#include <stdio.h>
int main(){

int num1, num2;

printf("Enter two numbers: ");/* Input numbers */
scanf("%d%d", &num1, &num2);

printf("Before swapping in main n");/* Print original values of num1 and num2 */
printf("Value of num1 = %d \n", num1);
printf("Value of num2 = %d \n\n", num2);

swap(&num1, &num2);/* Pass the addresses of num1 and num2 */

printf("After swapping in main n");/* Print the swapped values of num1 and num2 */
printf("Value of num1 = %d \n", num1);
printf("Value of num2 = %d \n\n", num2);
return 0;
}

void swap(int * num1, int * num2){

int temp;

temp = *num1;// Copy the value of num1 to some temp variable
*num1= *num2;// Copy the value of num2 to num1
*num2= temp;// Copy the value of num1 in temp to num2

}
```



```
Enter two numbers: 5
10
Before swapping in main nValue of num1 = 5
Value of num2 = 10

After swapping in main nValue of num1 = 10
Value of num2 = 5
```

THANK YOU !

ICT 131-3

Uva Wellassa University

Practical 7 - C Programming VII