Formal Languages and Compilers

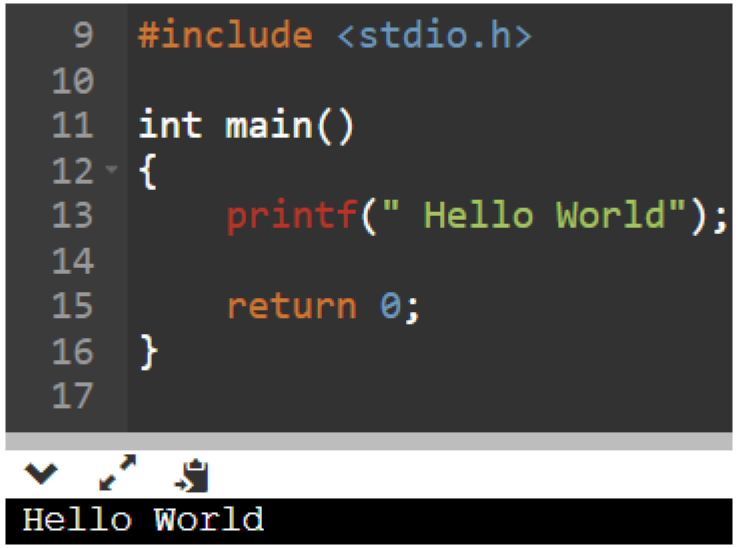
Lab8

Introduction to C

- C is a procedural programming language(=function based):
  - the program is divided into small parts called ***functions***
  - overloading is not possible
  - there are no objects, no classes
- Suitable for system programming like operating system or compiler development
- Compiled language (middle-level language)
- Used to create: Windows, Python, Git

Syntax

- #include <stdio.h> -header file library used to be able to work with input/output functions
- main() – function; anything inside {} will be executed
- Every statement ends with ;
- printf() – function to print (output)
- return – end of a function



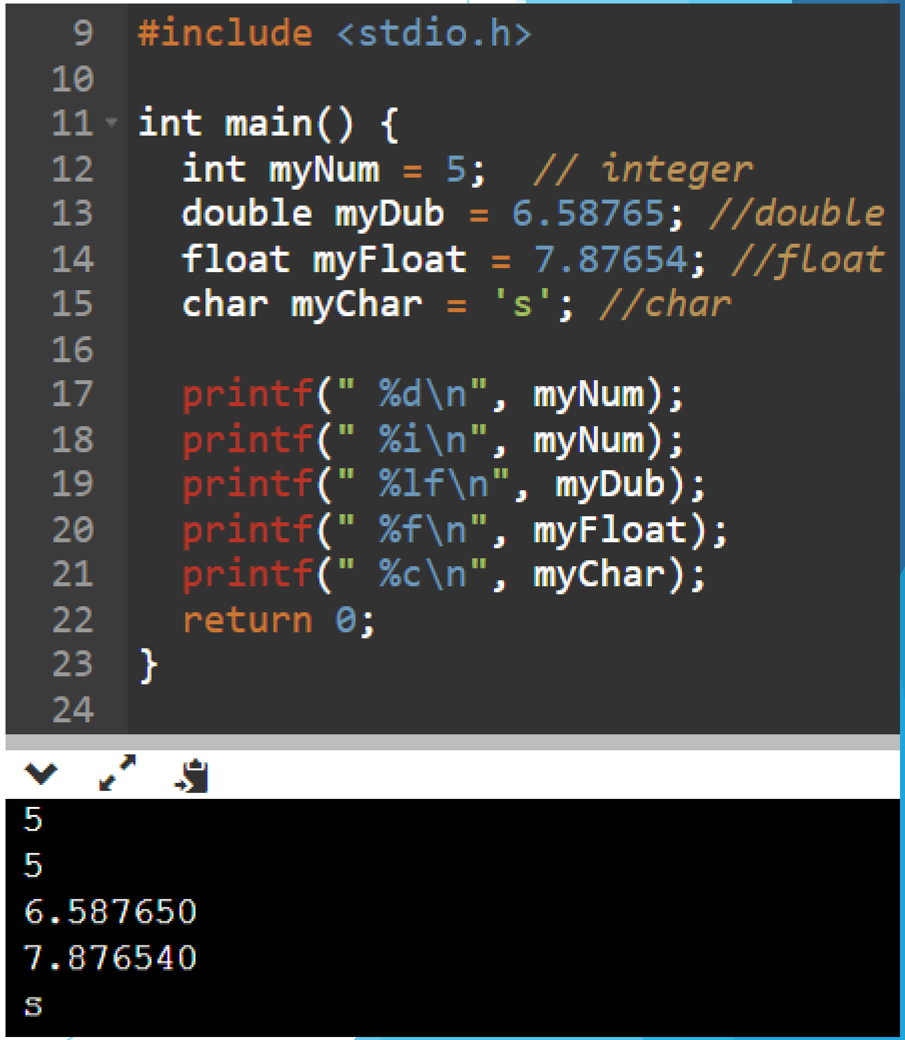
```
#include <stdio.h>

int main()
{
    printf(" Hello World");

    return 0;
}

```

```
Hello World
```

Data Types

Data Type	Format Specifier	Description
int	%i or %d	Store numbers (without decimals)
float	%f	Store fractional values (up to 7 decimals)
double	%lf	Store fractional values (up to 15 decimals)
char	%c	Store a single character/letter/number/ASCII value

- Strings are not a native data type in C. In order to create a String variable, a list of characters must be used.
- To declare a variable, the following syntax must be used:
  type variableName = *value*;
- Format specifiers are used inside printf() to tell the compiler what data type is the variable storing.



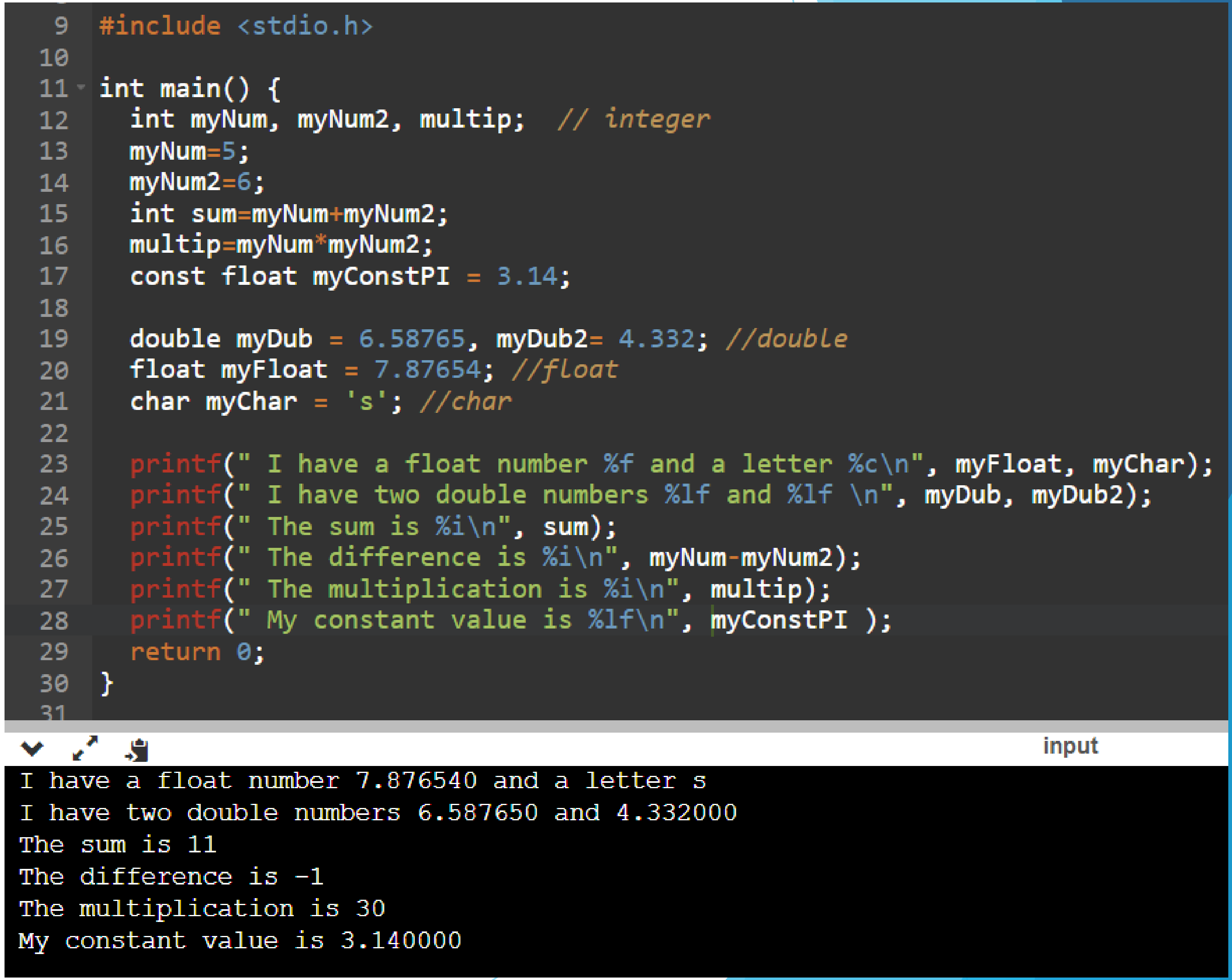
```c
#include <stdio.h>

int main() {
   int myNum = 5;  // integer
   double myDub = 6.58765; //double
   float myFloat = 7.87654; //float
   char myChar = 's'; //char

   printf(" %d\n", myNum);
   printf(" %i\n", myNum);
   printf(" %lf\n", myDub);
   printf(" %f\n", myFloat);
   printf(" %c\n", myChar);
   return 0;
}
```

```
5
5
6.587650
7.876540
s
```

Variables

- Names (=identifiers):
  - can contain letters, digits and underscores
  - Can start with a letter or an underscore
  - Are case sensitive
  - Cannot be the same as reserved words (e.g., int, float)

- **const** –keyword to declare constant variables (unchangeable, read-only)



```c
#include <stdio.h>

int main() {
  int myNum, myNum2, multip;  // integer
  myNum=5;
  myNum2=6;
  int sum=myNum+myNum2;
  multip=myNum*myNum2;
  const float myConstPI = 3.14;

  double myDub = 6.58765, myDub2= 4.332; //double
  float myFloat = 7.87654; //float
  char myChar = 's'; //char

  printf(" I have a float number %f and a letter %c\n", myFloat, myChar);
  printf(" I have two double numbers %lf and %lf \n", myDub, myDub2);
  printf(" The sum is %i\n", sum);
  printf(" The difference is %i\n", myNum-myNum2);
  printf(" The multiplication is %i\n", multip);
  printf(" My constant value is %lf\n", myConstPI );
  return 0;
}
```

```
I have a float number 7.876540 and a letter s
I have two double numbers 6.587650 and 4.332000
The sum is 11
The difference is -1
The multiplication is 30
My constant value is 3.140000
```

Operators

- Arithmetic operators

Operator	Name	Example int x=7, y=5	Result
+	Addition	x+y	12
-	Subtraction	x-y	2
*	Multiplication	x*y	35
/	Division	x/y	1
%	Modulus	x%y	2
++	Increment	++x	x=8
--	Decrement	--x	x=6

- Assignment operators

Operator	Example (int)	Meaning
=	x=10	x=10
+=	x+=3	x=x+3 =>x=13
-=	x-=3	x=x-3 => x=7
*=	x*=3	x=x*3 => x=30
/=	x/=3	x=x/3 => x=3
%=	x%=3	x=x%3 => x=1

Operators

- Comparison operators
  (The returned value is 1 for true and 0 for false)

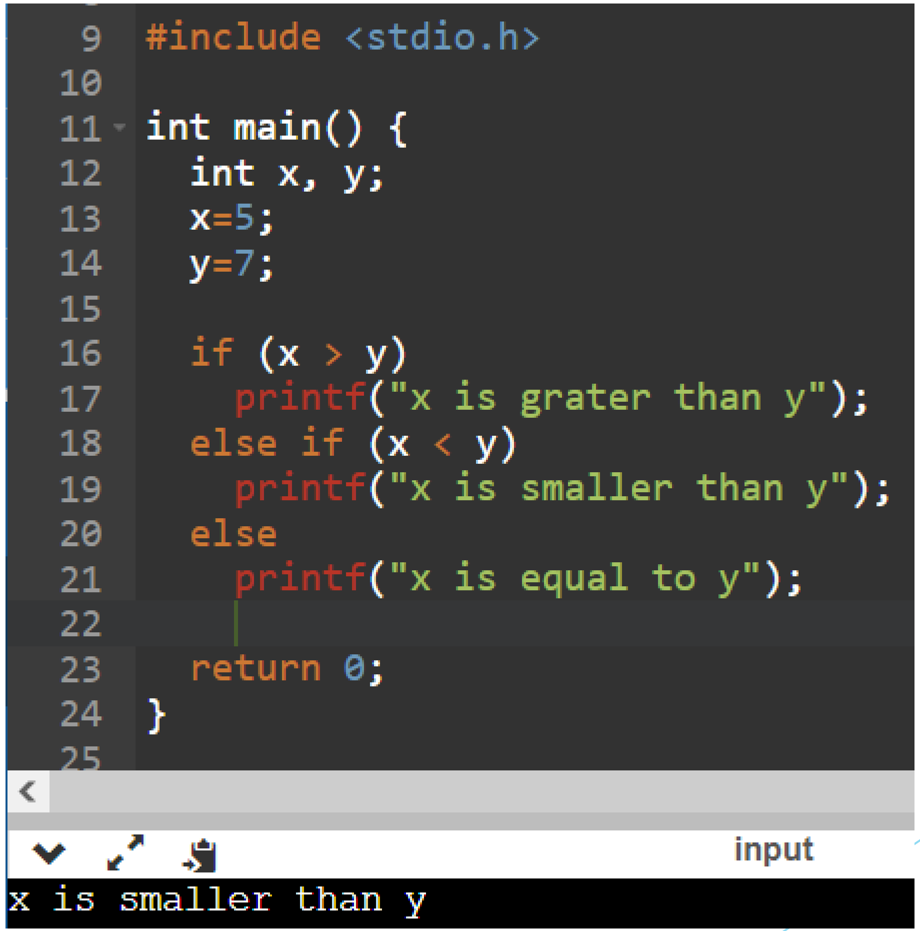
Operator	Name
==	Equal
!=	Not equal
>	Greater than
>	Less than
>=	Grater than or equal to
<=	Less than or equal to

- Logical operators
  (The returned value is 1 for true and 0 for false)

Operator	Meaning	Example x=7
&&	Logical AND - Returns True if both statements are true	x>2 && x<8 1 x>15 && x <20 0 x>10 && x <5 0
||	Logical OR - Returns True if one of the statements is true	x>2 || x<8 1 x>15 || x <20 1 x>10 || x <5 0
!	Logical NOT - Reverse the result (if the result is True, it returns False)	!(x>2 && x<8) 0 !(x>15 && x <20) 1 !(x>10 && x <5) 1 !(x>2 || x<8) 0 !(x>15 || x <20) 0 !(x>10 || x <5) 1

If…else if…else

if (*condition1*) {
  // block of code to be executed if condition1 is true
} else if (*condition2*) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}



```
#include <stdio.h>

int main() {
  int x, y;
  x=5;
  y=7;

  if (x > y)
    printf("x is grater than y");
  else if (x < y)
    printf("x is smaller than y");
  else
    printf("x is equal to y");

  return 0;
}
```

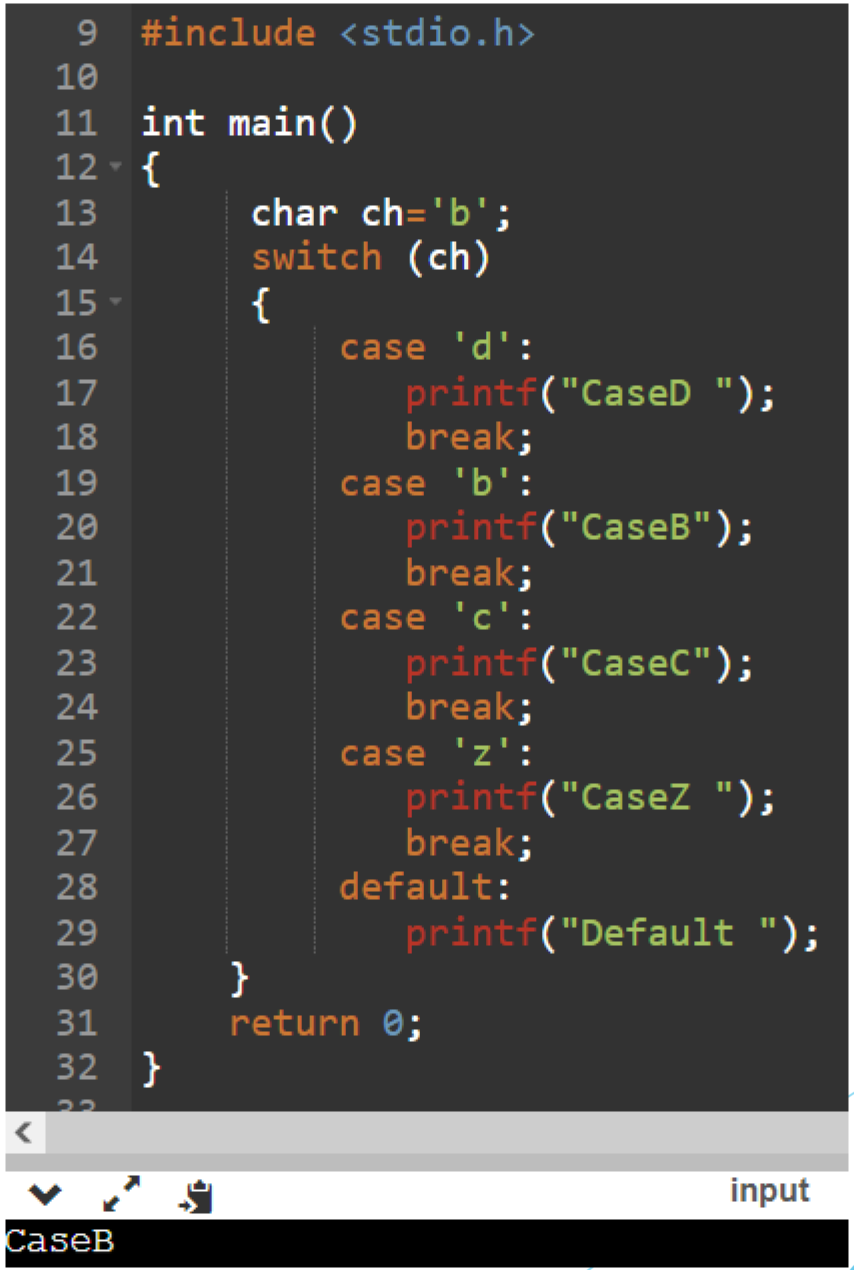
```
x is smaller than y
```

Switch

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

default-optional (runs if no case match)

Duplicate cases are not allowed



```c
#include <stdio.h>

int main()
{
        char ch='b';
        switch (ch)
        {
            case 'd':
                printf("CaseD ");
                break;
            case 'b':
                printf("CaseB");
                break;
            case 'c':
                printf("CaseC");
                break;
            case 'z':
                printf("CaseZ ");
                break;
            default:
                printf("Default ");
        }
       return 0;
}
```

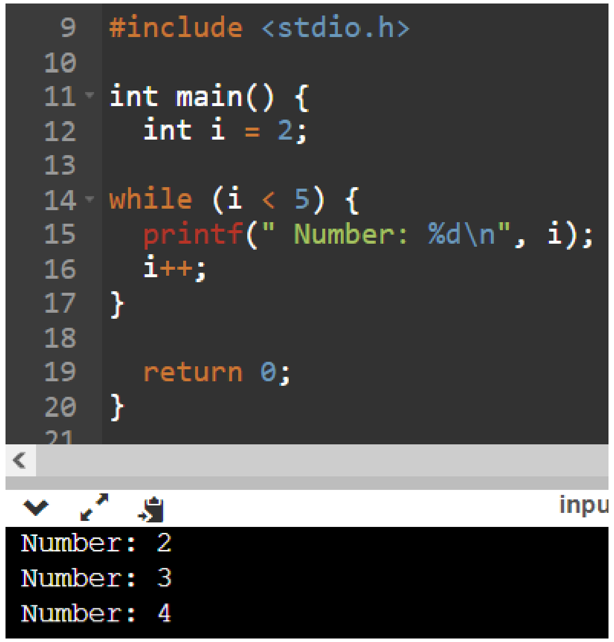
```
CaseB
```

While / do…while loop

```
while (condition) {
  // code block to be executed
}
```

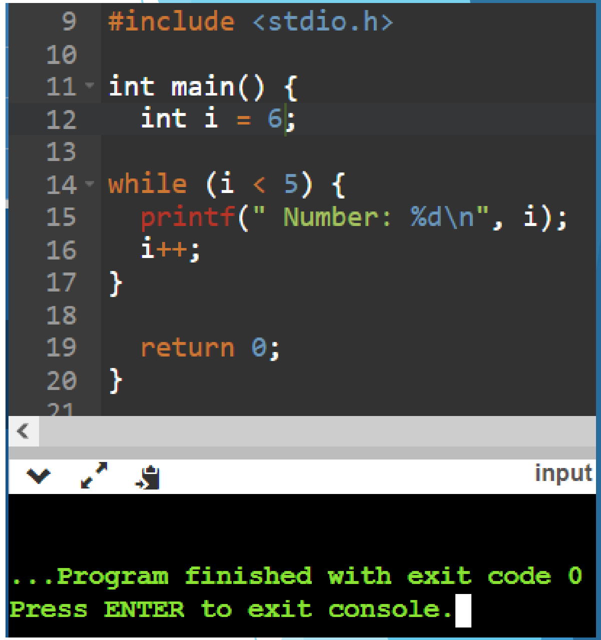
```
do {
  // code block to be executed
}
while (condition);
```

-the code will execute at least once, even if the condition is not true because the block code is executed before verifying the condition



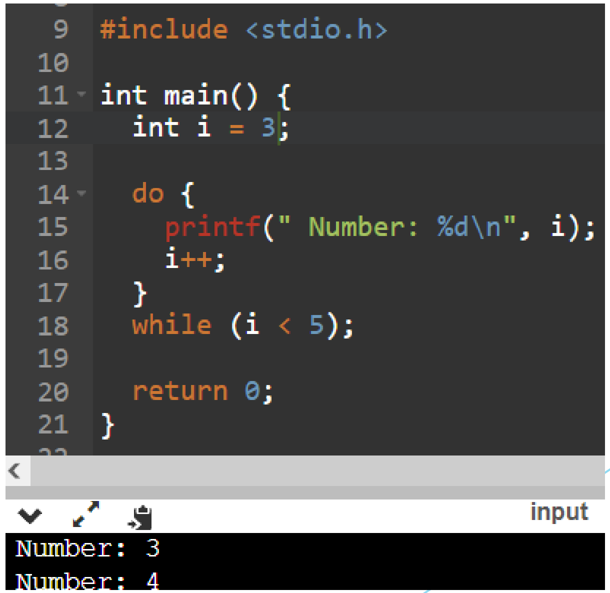
```c
#include <stdio.h>

int main() {
  int i = 2;

while (i < 5) {
  printf(" Number: %d\n", i);
  i++;
}

  return 0;
}
```

```
Number: 2
Number: 3
Number: 4
```



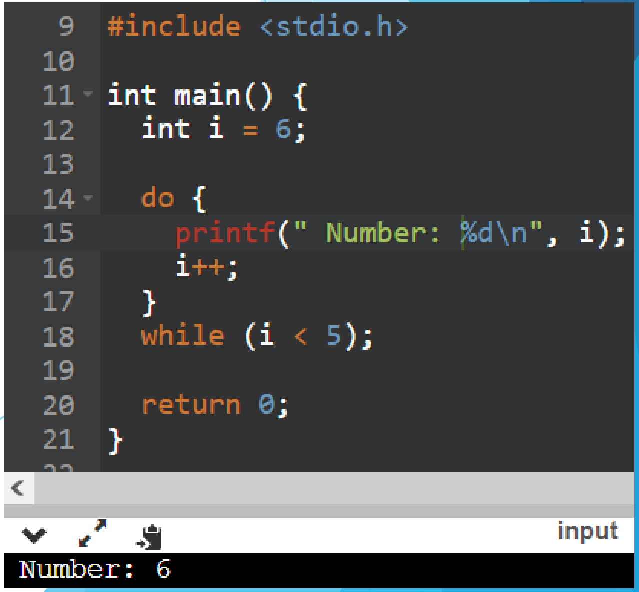
```c
#include <stdio.h>

int main() {
  int i = 6;

while (i < 5) {
  printf(" Number: %d\n", i);
  i++;
}

  return 0;
}
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```



```c
#include <stdio.h>

int main() {
  int i = 3;

  do {
    printf(" Number: %d\n", i);
    i++;
  }
  while (i < 5);

  return 0;
}
```

```
Number: 3
Number: 4
```



```c
#include <stdio.h>

int main() {
  int i = 6;

  do {
    printf(" Number: %d\n", i);
    i++;
  }
  while (i < 5);

  return 0;
}
```

```
Number: 6
```

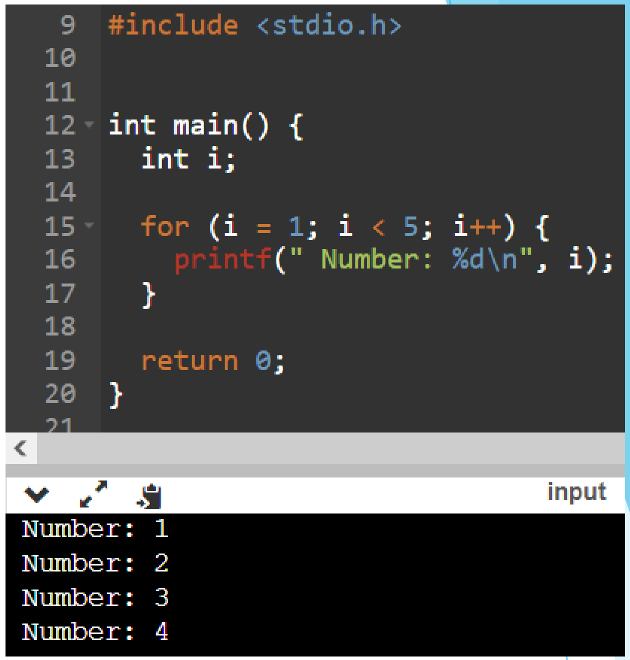
For loop

<span style="color:blue">for</span> (*statement 1*; *statement 2*; *statement 3*) {
  // code block to be executed
}

- **Statement 1** is executed (one time) before the execution of the code block.
- **Statement 2** defines the condition for executing the code block.
- **Statement 3** is executed (every time) after the code block has been executed.

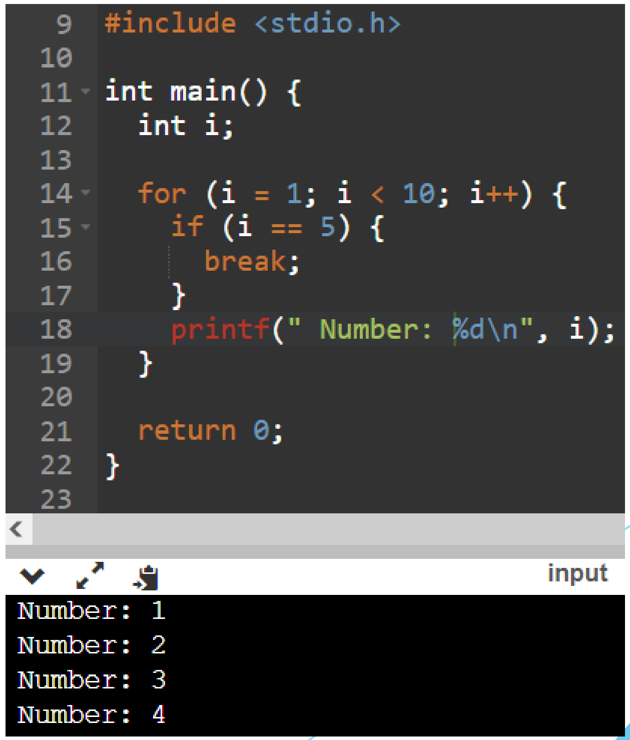
continue - breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

break - used to jump out of a **loop**.



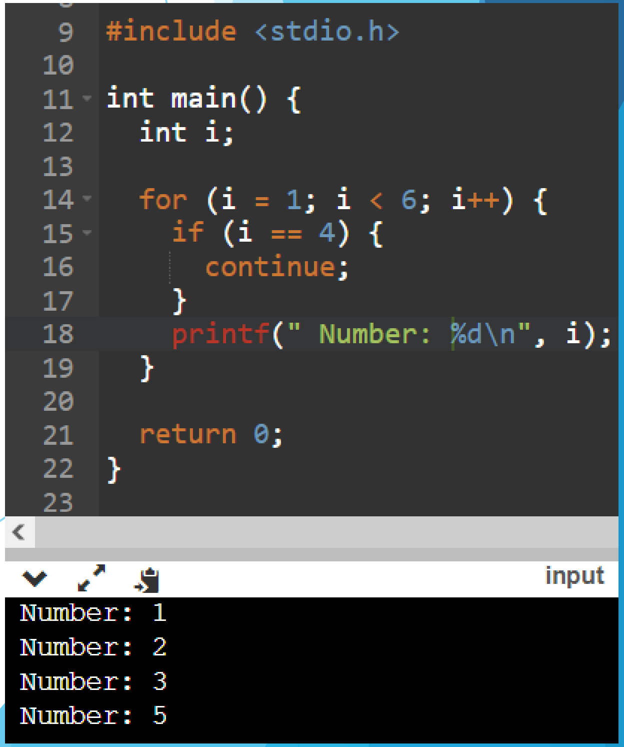
```c
#include <stdio.h>


int main() {
  int i;

  for (i = 1; i < 5; i++) {
    printf(" Number: %d\n", i);
  }

  return 0;
}
```

```
Number: 1
Number: 2
Number: 3
Number: 4
```



```c
#include <stdio.h>

int main() {
  int i;

  for (i = 1; i < 10; i++) {
    if (i == 5) {
      break;
    }
    printf(" Number: %d\n", i);
  }

  return 0;
}
```

```
Number: 1
Number: 2
Number: 3
Number: 4
```



```c
#include <stdio.h>

int main() {
  int i;

  for (i = 1; i < 6; i++) {
    if (i == 4) {
      continue;
    }
    printf(" Number: %d\n", i);
  }

  return 0;
}
```

```
Number: 1
Number: 2
Number: 3
Number: 5
```

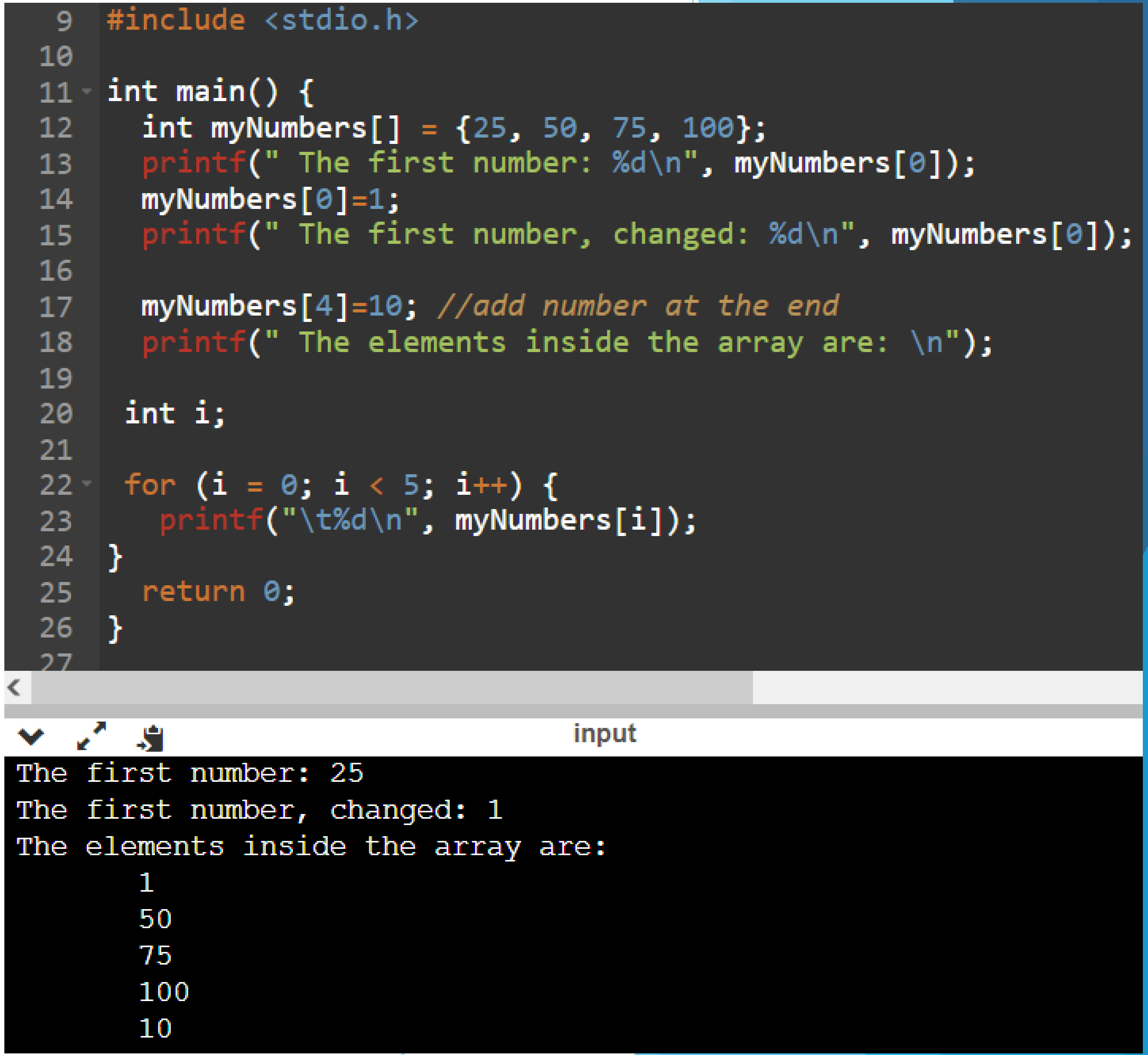
Arrays

- store multiple values in a single variable

- syntax:

datatype name[] = {value1,value2..}

- access elements inside the array: index



```c
#include <stdio.h>

int main() {
  int myNumbers[] = {25, 50, 75, 100};
  printf(" The first number: %d\n", myNumbers[0]);
  myNumbers[0]=1;
  printf(" The first number, changed: %d\n", myNumbers[0]);

  myNumbers[4]=10; //add number at the end
  printf(" The elements inside the array are: \n");

 int i;

 for (i = 0; i < 5; i++) {
   printf("\t%d\n", myNumbers[i]);
}
  return 0;
}
```

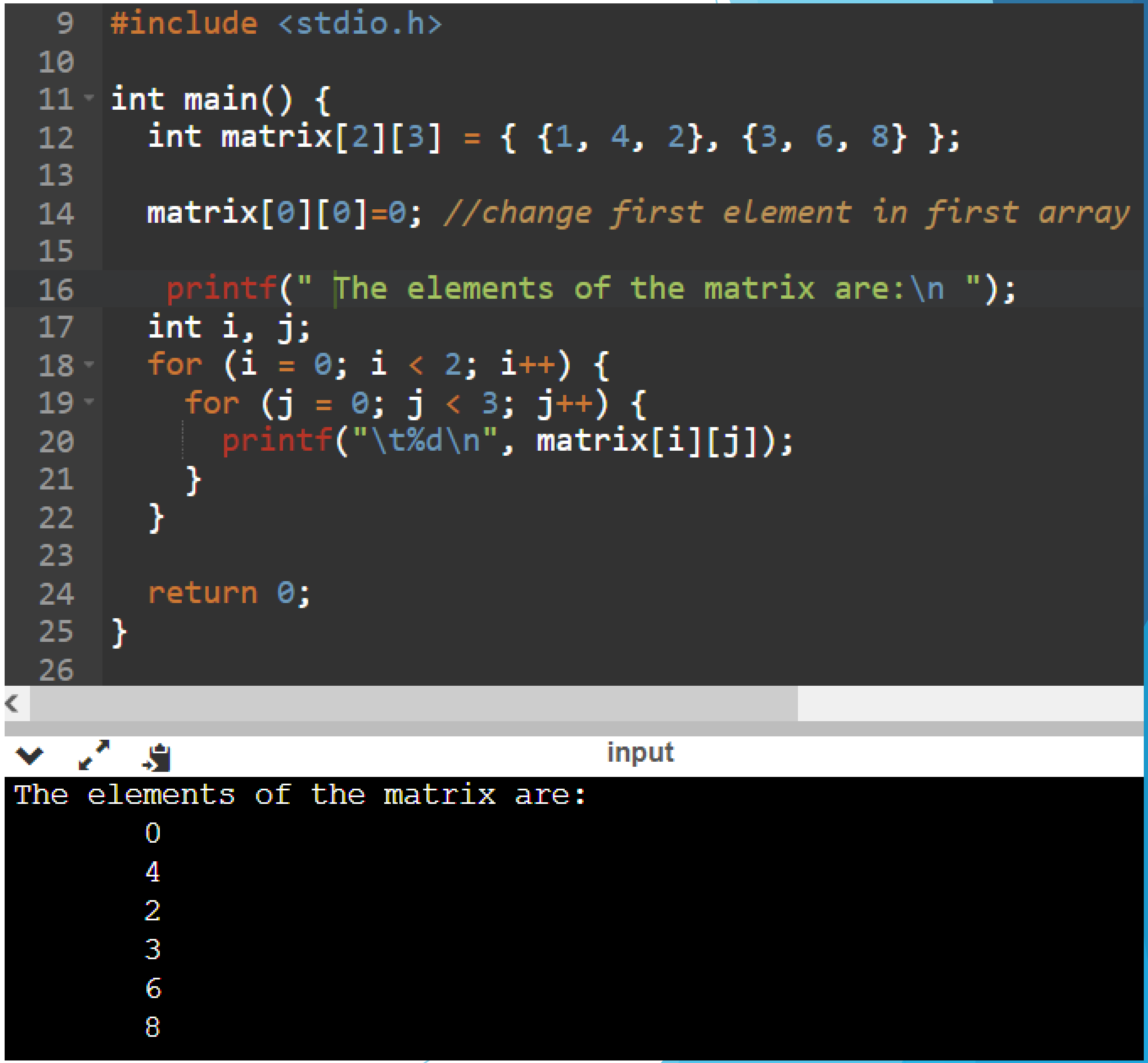
input

```
The first number: 25
The first number, changed: 1
The elements inside the array are:
        1
        50
        75
        100
        10
```

2D arrays

- Array of arrays (known as a matrix)
- Syntax:

datatype name[no_rows][no_columns] = {{value11,value12..},{value21,value22..}}

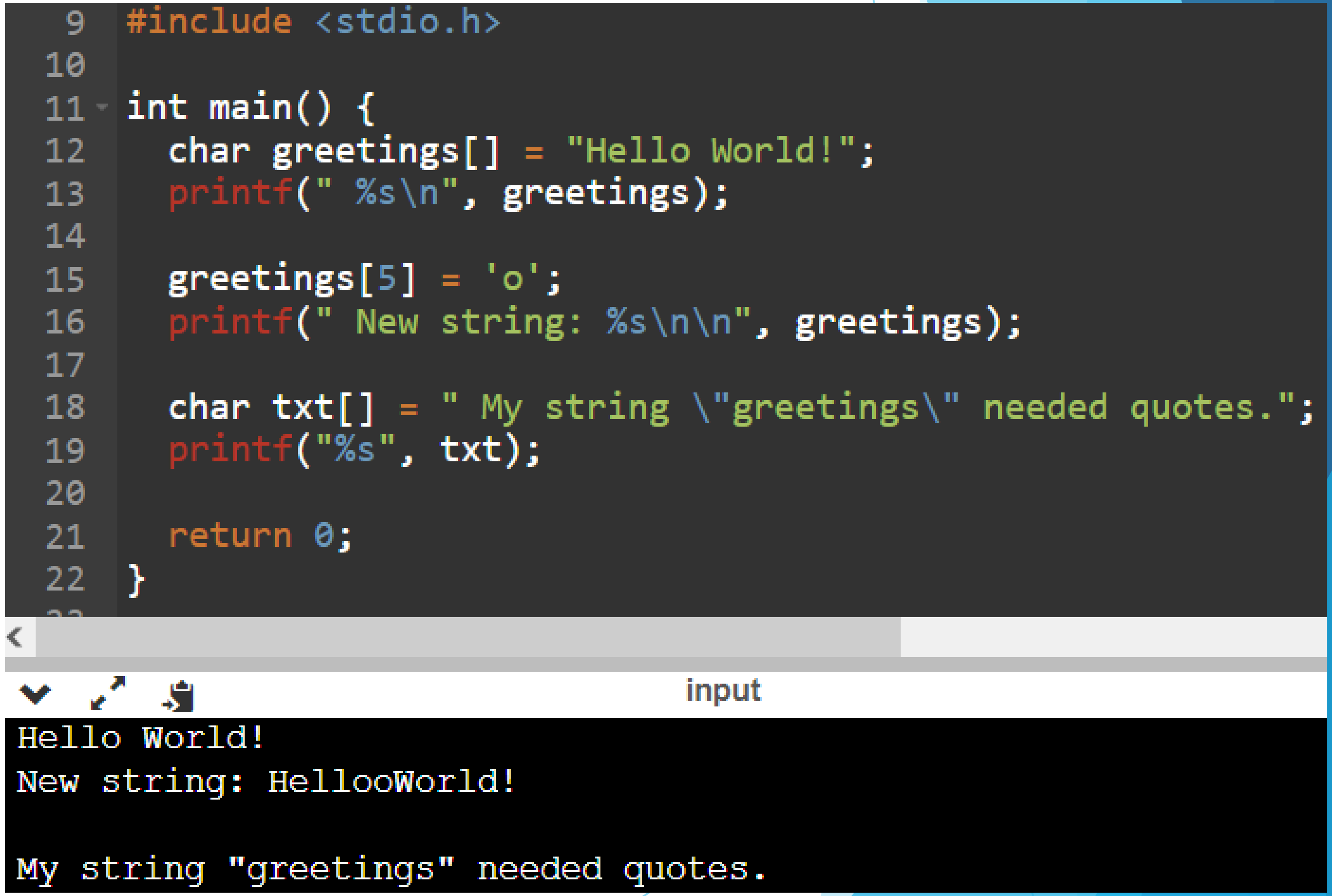


```c
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

  matrix[0][0]=0; //change first element in first array

   printf(" The elements of the matrix are:\n ");
  int i, j;
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
      printf("\t%d\n", matrix[i][j]);
    }
  }

  return 0;
}

```

```
The elements of the matrix are:
        0
        4
        2
        3
        6
        8
```

Strings

- Syntax: **char name[] = ” ”**
- mandatory double quotes for the value of the string
- Format specifier: %s
- Access elements inside the String: index
- Use \’ or \" special characters to add quotation marks inside a string



```c
#include <stdio.h>

int main() {
  char greetings[] = "Hello World!";
  printf(" %s\n", greetings);

  greetings[5] = 'o';
  printf(" New string: %s\n\n", greetings);

  char txt[] = " My string \"greetings\" needed quotes.";
  printf("%s", txt);

  return 0;
}
```

input

```
Hello World!
New string: HellooWorld!

My string "greetings" needed quotes.
```

Exercises

- Go to https://www.onlinegdb.com/online_c_compiler

1. Check if a year is a leap year. (leap years are divisible by 4 and not divisible by 100 or divisible by 400)
2. Using switch, find if a number is positive, negative or 0.
3. Find the sum of all the odd numbers between 0 and 20.
4. Print the number of digits in an integer (e.g., x=123456, no_digits=6)
5. Find the sum of all elements in an array (e.g., arr[]={1,2,3,4,5} sum=15)
6. Find the maximum between 3numbers.

Homework

- Install Code::Blocks

1. Search if an element is in an array. (e.g., arr[]={1,2,3,4,5} x=5 x is in arr)
2. Add two 2D arrays. (e.g., arr[2][2] ={{1,2},{3,4}} arr2[2][2] ={{5,6},{7,8}} sum[2][2]={{5,8},{10,12}}
3. Check if two 2D arrays are equal.(e.g., arr[2][2] ={{1,2},{3,4}} arr2[2][2] ={{1,2},{3,4}} are equal)
4. Find the length of a string. (e.g., char str[]="Hello" length=5)
5. Search for the word FILS and display its index in the following array: char ex[] = "FILS is part of UPB. I am a student at FILS. Welcome!"
6. Check if a string is a palindrome.