C Syntax

=> sebuah aturan yang digunakan untuk menulis kalimat agar mampu dimengerti oleh bahasa pemrograman. Dalam pembuatannya, seluruh aturan syntax harus terpenuhi. Karena ketika proses kompilasi setiap barus script akan dilakukan pengecekan. Jika terdapat syntax yang salah maka compiler akan melaporan terjadinya error massage dan tidak akan meneruskan pembuatan bytecodenya.

```
1  #include <stdio.h>
2
3  int main() {
4    printf("Hello World!");
5    return 0;
6  }
```

```
C:\Users\User\Downloads\syntax.exe
Hello World!
--------------------------------
Process exited after 0.03957 seconds with return value 0
Press any key to continue . . .
```

Exercise:

Insert the missing part of the code below to output "Hello World!".

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

C Output

=> hasil program yang telah dicompile

```
1  #include <stdio.h>
2
3  int main() {
4    printf("Hello World!");
5    printf("I am learning C.");
6    return 0;
7  }
```

```
C:\Users\User\Downloads\output.exe
Hello World!I am learning C.
--------------------------------
Process exited after 0.09477 seconds with return value 0
Press any key to continue . . .
```

C Comments

=> untuk mendeskripsikan sebuah kode yang telah dibuat atau supaya code lebih mudah dibaca

```c
#include <stdio.h>
int main()
{
    printf("Hello World"); //ini merupakan comment
    return 0;
}
```

```
C:\Users\User\Downloads\comment.exe
Hello World
--------------------------------
Process exited after 0.08101 seconds with return value 0
Press any key to continue . . .
```

Exercise:

Comments in C are written with special characters. Insert the missing parts:

```c
// This is a single-line comment
/* This is a multi-line comment */
```

C Variables

=> Variabel adalah penanda identitas yang digunakan untuk menampung suatu nilai.

```c
#include <stdio.h>

int main() {
  // Create variables
  int myNum = 5;               // Integer (whole number)
  float myFloatNum = 5.99;     // Floating point number
  char myLetter = 'D';         // Character

  // Print variables
  printf("%d\n", myNum);
  printf("%f\n", myFloatNum);
  printf("%c\n", myLetter);
  return 0;
}
```

```
C:\Users\User\Downloads\variables.exe
5
5.990000
D

--------------------------------
Process exited after 0.01375 seconds with return value 0
Press any key to continue . . .
```

Exercise:

Create a variable named `myNum` and assign the value `50` to it.

```c
int myNum = 50;
```

Exercise:

Use the correct **format specifier** to output the value of `myNum` :

```
int myNum = 15;
printf("%d", myNum);
```

C Data Types

=> Tipe data atau kadang disingkat dengan ‘tipe’ saja adalah sebuah pengelompokan data untuk memberitahu compiler atau interpreter bagaimana programmer ingin mengolah data tersebut.

```c
#include <stdio.h>

int main() {
  // Create variables
  int myNum = 5;               // Integer (whole number)
  float myFloatNum = 5.99;     // Floating point number
  char myLetter = 'D';         // Character

  // Print variables
  printf("%d\n", myNum);
  printf("%f\n", myFloatNum);
  printf("%c\n", myLetter);
  return 0;
}
```

```
C:\Users\User\Downloads\data types.exe
5
5.990000
D

--------------------------------
Process exited after 0.1124 seconds with return value 0
Press any key to continue . . .
```

Exercise:

Display the sum of `5 + 10`, using two variables: `x` and `y`.

```
int x = 5;
int y = 10;
printf("%d", x + y);
```

Exercise:

Fill in the missing parts to create three variables of the same type, using a **comma-separated list**:

```
int x = 5, y = 6, z = 50;
printf("%d", x + y + z);
```

Exercise:

Add the correct data type for the following variables:

```
int myNum = 5;
float myFloatNum = 5.99;
char myLetter = 'D';
```

Exercise:

Add the correct format specifier to print the value of the following variable:

```
char myLetter = 'D';
printf("%c", myLetter);
```

C Constants

1) Agar variabel yang sudah ada tidak dapat dirubah-rubah oleh sembarang orang maupun diri kita sendiri, kita bisa menggunakan keyword `const`.

```c
#include <stdio.h>

int main(){
    const int myNum = 15;
    myNum = 10;

    printf("%d", myNum);
    return 0;
}
```

```
PS D:\myfile\KULIAH\Semester 5\4-PARADIGMA PEMROGRAMAN\C> cd "d:\myfile\KULIA
H\Semester 5\4-PARADIGMA PEMROGRAMAN\C\" ; if ($?) { gcc test.c -o test } ; i
f ($?) { .\test }
test.c: In function 'main':
test.c:5:11: error: assignment of read-only variable 'myNum'
    5 |     myNum = 10;
      |           ^
```

2) Biasanya dipakai untuk nilai yang tidak mungkin berubah seperti *phi,*

```c
#include <stdio.h>

int main(){
    const int minutesPerHour = 60;
    const float PI = 3.14;

    printf("%d\n", minutesPerHour);
    printf("%f\n",PI);
    return 0;
}
```

```
PS D:\myfile\KULIAH\Semester 5\4-PARADIGMA PEMROGRAMAN\C> cd "d:\myfile\KULIA
H\Semester 5\4-PARADIGMA PEMROGRAMAN\C\" ; if ($?) { gcc test.c -o test } ; i
f ($?) { .\test }
60
3.140000
```

C Operators

```c
int main(){
    int sum1 = 100+50;      // 150 (100 + 50)
    int sum2 = sum1+250;    // 400 (150 + 250)
    int sum3 = sum2+sum2;   // 800 (400 + 400)
    printf("%d\n", sum1);
    printf("%d\n", sum2);
    printf("%d\n", sum3);
    return 0;
}
```

```
PS D:\myfile\KULIAH\Semester 5\4-PARADIGMA PEMROGRAMAN\C> c
d "d:\myfile\KULIAH\Semester 5\4-PARADIGMA PEMROGRAMAN\C\"
; if ($?) { gcc test.c -o test } ; if ($?) { .\test }
150
400
800
```

Exercise:

Fill in the blanks to multiply `10` with `5`, and print the result.

```c
int x = 10;
int y = 5;
printf("%d", x * y);
```

Exercise:

Fill in the blanks to divide `10` by `5`, and print the result.

```
int x = 10;
int y = 5;
printf("%d", x / y);
```

Use the correct operator to increase the value of the variable `x` by `1`.

```
int x = 10;
x++;
```

Use the **addition assignment** operator to add the value `5` to the variable `x`.

```
int x = 10;
x += 5;
```

C If...Else

```c
int main(){
    int x = 20;
    int y = 18;
    if (x>y)
    {
        printf("x is greater than y");
    }
    return 0;
}
```

```
PS D:\myfile\KULIAH\Semester 5\4-PARADIGMA PEMROGRAMAN\C> c
d "d:\myfile\KULIAH\Semester 5\4-PARADIGMA PEMROGRAMAN\C\"
; if ($?) { gcc test.c -o test } ; if ($?) { .\test }
x is greater than y
```

Exercise:

Print "Hello World" if `x` is **greater than** `y`.

```
int x = 50;
int y = 10;
if (x > y) {
  printf("Hello World");
}
```

Exercise:

Print "Hello World" if `x` is **equal to** `y`.

```
int x = 50;
int y = 50;
if (x == y) {
  printf("Hello World");
}
```

Exercise:

Print "Yes" if `x` is equal to `y`, otherwise print "No".

```
int x = 50;
int y = 50;
if (x == y) {
  printf("Yes");
} else {
  printf("No");
}
```

C Switch

```c
#include <stdio.h>

int main() {
  int day = 4;

  switch (day) {
    case 1:
      printf("Monday");
      break;
    case 2:
      printf("Tuesday");
      break;
    case 3:
      printf("Wednesday");
      break;
    case 4:
      printf("Thursday");
      break;
    case 5:
      printf("Friday");
      break;
    case 6:
      printf("Saturday");
      break;
    case 7:
      printf("Sunday");
      break;
  }

  return 0;
}
```

```
d "d:\myfile\KULIAH\Semester 5\4-PARADIGMA PEMROGRAMAN\C\"
; if ($?) { gcc test.c -o test } ; if ($?) { .\test }
Thursday
```

Exercise:

Insert the missing parts to complete the following `switch` statement.

```
int day = 2;
switch (day) {
  case 1:
    printf("Monday");
    break;
  case 2:
    printf("Sunday");
    break;
}
```

Exercise:

Complete the `switch` statement, and add the correct **keyword** at the end to specify some code to run if there is no case match in the `switch` statement.

```
int day = 4;
switch (day) {
  case 1:
    printf("Saturday");
    break;
  case 2:
    printf("Sunday");
    break;
  default:
    printf("Weekend");
}
```

C While Loop

=> loop dapat mengeksekusi blok kode selama kondisi tertentu tercapai, loop berguna karena menghemat waktu, mengurangi kesalahan, dan membuat kode lebih mudah dibaca.

Exercise 1

Exercise:

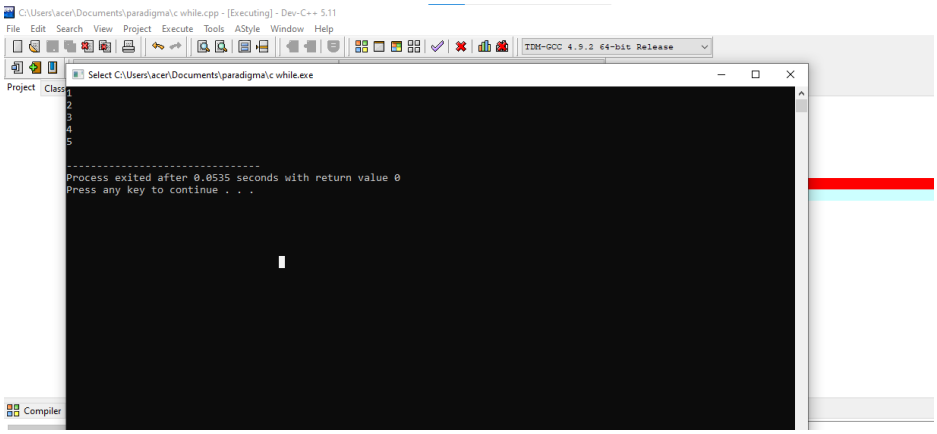
Print `i` as long as `i` is less than 6.

```
int i = 1;
while (i < 6) {
  printf("%d\n", i);
  i++;
}
```

Show Answer

Submit Answer ›

Exercise 2



C For Loop

Exercise:

Print `i` as long as `i` is less than 6.

```
int i = 1;
while (i < 6) {
  printf("%d\n", i);
  i++;
}
```

Exercise:

Use the `do/while` loop to print `i` as long as `i` is less than 6.

```
int i = 1;
do {
  printf("%d\n", i);
  i++;
}
while (i < 6);
```

Exercise:

Use a **for loop** to print "Yes" 5 times:

```
for (int i = 0; i < 5; i++) {
  printf("Yes\n");
}
```

C Break/Continue

Exercise:

Stop the loop if `i` is 5.

```
for (int i = 0; i < 10; i++) {
  if (i == 5) {
    break;
  }
  printf("%d\n", i);
}
```

Exercise:

In the following loop, when the value is "4", jump directly to the next value.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  printf("%d\n", i);
}
```

C Arrays

Exercise:

Create an array of type `int` called `myNumbers`.

```
int myNumbers[] = {25, 50, 75, 100};
```

Exercise:

Print the value of the second element in the `myNumbers` array.

```
int myNumbers[] = {25, 50, 75, 100};
printf("%d", myNumbers[1]);
```

Exercise:

Change the value of the first element to 33:

```
int myNumbers[] = {25, 50, 75, 100};
myNumbers[0] = 33;
```

Exercise:

Loop through the elements in the array using a **for loop**.

```
int myNumbers[] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", myNumbers[i]);
}
```

C String

string

Untitled3.c

```
#include <stdio.h>

int main() {
  char greetings[] = "Hello World!";
  printf("%s", greetings);
  return 0;
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
Hello World!
--------------------------------
Process exited after 0.03961 seconds with return value 0
Press any key to continue . . .
```

Untuk mengubah isi dari string

Untitled3.c

```
#include <stdio.h>

int main() {
  char greetings[] = "Hello World!";
  greetings[2] = 'J';
  printf("%s", greetings);
  return 0;
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
HeJlo World!
--------------------------------
Process exited after 0.02602 seconds with return value 0
Press any key to continue . . .
```

EXERCISE

Exercise:

Fill in the missing part to create a "string" named **greetings**, and assign it the value "Hello".

```
char greetings[] = "Hello";
```

Exercise:

Another way of creating strings:
Fill in the missing part to create a "string" named **greetings**, and assign it the value "Hi".

```
char greetings[] = {'H', 'i', "\0"};
```

Exercise:

Use the correct format specifier to output the string:

```
char greetings[] = "Hello World!";
printf("%s", greetings);
```

Exercise:

Print the first character in the greetings string:

```
char greetings[] = "Hello World!";
printf("%c", greetings[0]);
```

Exercise:

Use the correct format specifier to output the string:

```
char greetings[] = "Hello World!";
printf("%s", greetings);
```

C User Input

Input user int

Untitled3.c

```
#include <stdio.h>

int main() {
  int me;
  printf("ketik nomor dan enter\n");
  scanf("%d", &me);
  printf("nomor kamu adalah : %d", me);
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
ketik nomor dan enter
1
nomor kamu adalah : 1
--------------------------------
Process exited after 2.888 seconds with return value 21
Press any key to continue . . .
```

Input user char (String)

Untitled3.c

```c
#include <stdio.h>

int main() {
  int me[30];
  printf("masukkan nama : \n");
  scanf("%s", &me);
  printf("nomor kamu adalah : %s", me);
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
masukkan nama :
varis_gito
nomor kamu adalah : varis_gito
--------------------------------
Process exited after 5.476 seconds with return value 30
Press any key to continue . . .
```

C Memory Addres

Untitled3.c

```c
#include <stdio.h>
int main(){
    int me = 19;
    printf("%p", &me);
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
000000000062FE1C
--------------------------------
Process exited after 0.02919 seconds with return value 16
Press any key to continue . . .
```

C Pointer

Untitled3.c

```c
#include <stdio.h>
int main(){
    int myAge = 43;
    int* ptr = &myAge;
    printf("%d\n", myAge);
    printf("%p\n", &myAge);
    printf("%p\n", ptr);
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
43
000000000062FE14
000000000062FE14

--------------------------------
Process exited after 0.03233 seconds with return value 17
Press any key to continue . . .
```

Exercise:

Create a pointer variable called **ptr**, that points to the `int` variable myAge:

```c
int myAge = 43;
int* ptr = &myAge;
```

C Function Paramater

1.

Untitled3.c

```c
#include <stdio.h>
void coba(char name[]) {
  printf("Hello %s\n", name);
}

int main() {
  coba("Liam");
  coba("Jenny");
  coba("Anja");
  return 0;
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
Hello Liam
Hello Jenny
Hello Anja

--------------------------------
Process exited after 0.03527 seconds with return value 0
Press any key to continue . . .
```

2.

Untitled3.c

```c
#include <stdio.h>
void myFunction(char name[], int age) {
  printf("Hello %s. You are %d years old.\n", name, age);
}
int main() {
  myFunction("Liam", 3);
  myFunction("Jenny", 14);
  myFunction("Anja", 30);
  return 0;
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
Hello Liam. You are 3 years old.
Hello Jenny. You are 14 years old.
Hello Anja. You are 30 years old.

--------------------------------
Process exited after 0.02892 seconds with return value 0
Press any key to continue . . .
```

3.

Untitled3.c

```c
#include <stdio.h>
int myFunction(int x) {
  return 5 + x;
}

int main() {
  printf("Result is: %d", myFunction(3));
  return 0;
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
Result is: 8
--------------------------------
Process exited after 0.02781 seconds with return value 0
Press any key to continue . . .
```

4.

Untitled3.c

```c
#include <stdio.h>
int myFunction(int x, int y) {
  return x + y;
}

int main() {
  printf("Result is: %d", myFunction(5, 3));
  return 0;
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled3.exe

```
Result is: 8
--------------------------------
Process exited after 0.02893 seconds with return value 0
Press any key to continue . . .
```

Exercise:

Create a method named `myFunction` and call it inside `main()`.

```c
void myFunction() {
  printf("I just got executed!");
}

int main() {
  myFunction();
  return 0;
}
```

Exercise:

Insert the missing parts to call `myFunction` **two times**.

```c
void myFunction() {
  printf("I just got executed!");
}

int main() {
  myFunction();
  myFunction();
  return 0;
}
```

Exercise:

Add a `name` parameter of type `char` (string) to `myFunction`.

```
void myFunction(char name[]) {
  printf("Hello %s\n", name);
}

int main() {
  myFunction("Liam");
  myFunction("Jenny");
  myFunction("Anja");
  return 0;
}
```

Exercise:

Insert the missing part to print the number **8** in `main`, by using a specific **keyword** inside `myFunction`:

```
int myFunction(int x) {
  return 5 + x;
}

int main() {
  printf("%d", myFunction(3));
  return 0;
}
```

C Recursion

Untitled1.cpp

```
#include <stdio.h>
int sum(int k);

int main() {
  int result = sum(10);
  printf("%d", result);
  return 0;
}

int sum(int k) {
  if (k > 0) {
    return k + sum(k - 1);
  } else {
    return 0;
  }
}
```

D:\TUGAS\KULIAH\semester 3\paradigma\Untitled1.exe

```
55
--------------------------------
Process exited after 0.02271 seconds with return value 0
Press any key to continue . . .
```

C Structure

Exercise:

Fill in the missing part to create a Car structure:

```
struct Car {
  char brand[50];
  char model[50];
  int year;
};
```

Exercise:

Fill in the missing parts to create a struct variable of "Car" named "car1" inside `main`:

```
struct Car {
  char brand[50];
  char model[50];
  int year;
};

int main() {
  struct Car car1;
  return 0;
}
```

Exercise:

Fill in the missing parts to assign the following values to the car1 variable:

"BMW" to `brand`, "X5" to `model` and 1999 to `year`.

```
struct Car {
  char brand[50];
  char model[50];
  int year;
};

int main() {
  struct Car car1 = {"BMW", "X5", 1999};
  return 0;
}
```