

Introduction to IoT

School Year 2023-2024

Valsalice



Course Structure

1	Introduction and Basics	
2	Basic Data Types and Operators	
3	Control Structures Pt. 1	
4	Control Structures Pt. 2	NOV 7
5	Functions and Scope	NOV 14
6	Arrays and Strings	NOV 21
10	Preprocessor and Macros	DEC 19
11	Custom Data Types	JAN 9

7	Introduction to Contiki-NG and nRF52840	NOV 28
8	Sensing and Actuating with Contiki-NG	DEC 5
9	Basic Communication and Networking	DEC 12
12	Introduction to RPL and Network Routing	JAN 16
13	Challenges in Wireless Communication	JAN 23
14	Advanced Protocols: TSCH and 6TiSCH	JAN 30
15	Advanced Topics in Wireless Communication	FEB 6
16	Reliable Data Transfer Challenge	FEB 20 FEB 27 MAR 5

■ = Core Topics ■ = Optional Topics

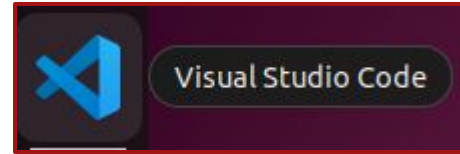
Open your Virtual Machines

1. Turn on your Laptops
2. Login using "User"
3. Open the **Virtual Box** program
4. Add the Virtual Machine (**Ctrl + A**)
5. Open the **VirtualBox** folder
6. Select the **nRF52840LAB** file
7. Click **Start**

Prepare the Coding Environment

1 Start the Virtual Machine **nRF52840LAB**

2 Open **Visual Studio Code**



3 From the Terminal:

```
make setup
```

4

```
➤ valsalice-iot-23 git:(master) make setup  
Enter your username: █
```

5

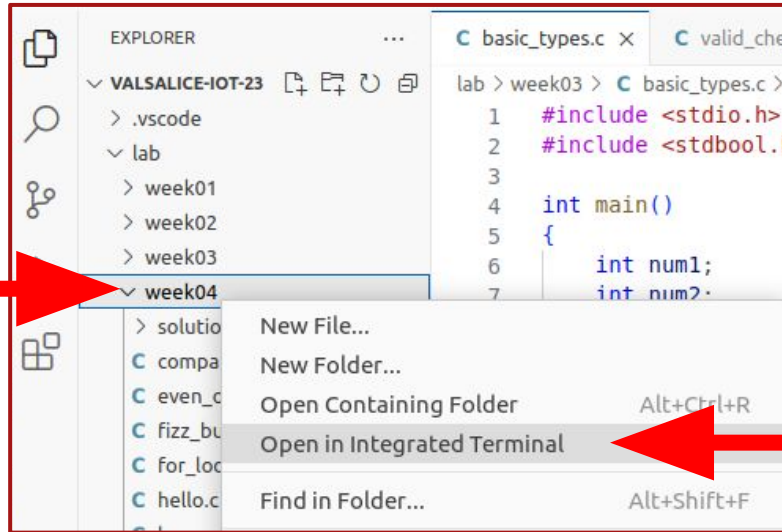
```
Password █
```

6

```
✓ Repository setup complete!
```

Prepare the Coding Environment

7 Open the **week04** folder in the terminal



8 You should see the following in the terminal:

The terminal screenshot shows the prompt 'week04 git:(master)' in a dark background with a red border.

Recap: Basic Input/Output Functions

- **printf**: C function for formatted output

```
printf("Hello, World!\n");
```

- **scanf**: C function for formatted input

```
char name[50];  
scanf("%s", name);
```

Recap: Data Types

C has a number of primitive data types:

int

42

1200

1_200

-3

float

3.14

0.00001

-2.1

char

'A'

'@'

'\n'

bool

true

false

Strings are *NOT* a primitive data type, and have special syntax.

strings

"Hello"

"A"

"I am a full sentence!"

Recap: Variables

A variable is a named container that stores data or values.

```
int x = 42;  
float y = -0.12;  
char w = 'A';  
char z[50] = "Full sentence";
```

Booleans require a custom include statement:

```
#include <stdbool.h>  
bool hello = true;
```


Recap: Comparisons

Values and variables can be compared.

```
int x = 10;  
int y = 4;  
  
bool w = x > y;  
printf("W: %d\n", w);  
  
bool z = x <= y;  
printf("Z: %d\n", z);
```

Recap: Boolean Operators

Greater than	>
Greater or equal than	>=
Less than	<
Less or equal than	<=
Equals	==
Not equals	!=
Not	!

Recap: Chaining Comparisons

- **and** (both must be true)

```
true && false
```

```
(5 < 6) && (5 < 10)
```

- **or** (either must be true)

```
true || false
```

```
(5 < 3) || (5 < 10)
```

- **not** (negation)

```
!true
```

```
!(5 < 3)
```

Recap: If-Statements

Allow for branches in your code!

```
int x = 5;

if (x < 10) {
    printf("X is small \n");
} else {
    printf("X is large \n");
}
```

```
int x = 20;

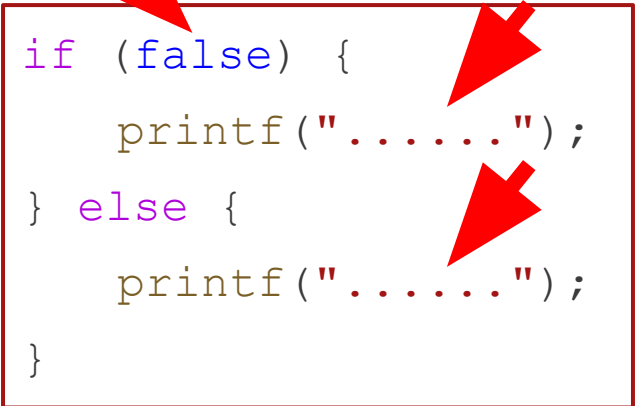
if (x < 10) {
    printf("X is small \n");
} else {
    printf("X is large \n");
}
```

NOTE: You do not need an else block, it's optional.

Exercise

Write a program (**even_odd.c**) that, given a variable **num**:

- Prints "EVEN" if the number is even.
- Prints "ODD" if the number is odd.



```
if (false) {  
    printf(".....");  
} else {  
    printf(".....");  
}
```

To execute:

make even_odd.run

Exercise - Solution

```
if (num % 2 == 0) {  
    printf("EVEN\n");  
} else {  
    printf("ODD\n");  
}
```

Save remotely your Changes

1

`make save`

2

`week04 git:(master) X make save`

3

`|Password``Git: https://aspina@git.spina.me (Press 'Enter' to confirm or
'Escape' to cancel)`

4

`✓ Changes committed and pushed. All done!`

If-Statement chaining

You can chain multiple conditions with **else if**.

What is the difference between these two snippets of code?

```
int num;
scanf("%d", &num);

if (num < 3) {
    printf("Small number\n");
} else if (num < 10) {
    printf("Medium number\n");
}
```

```
int num;
scanf("%d", &num);

if (num < 3) {
    printf("Small number\n");
}
if (num < 10) {
    printf("Medium number\n");
}
```


Exercise

Write a program (**comparison.c**) that, given a variable **num**:

- Prints “ZERO” if the number is equal to 0.
- Prints “POSITIVE” if the number is greater than 0.
- Prints “NEGATIVE” if the number is less than 0.

To execute:

```
make comparison.run
```

Exercise - Solution

```
if (num < 0) {  
    printf("NEGATIVE\n");  
} else if (num > 0) {  
    printf("POSITIVE\n");  
} else {  
    printf("ZERO\n");  
}
```

Save remotely your Changes

1

```
make save
```

2

```
week04 git:(master) X make save
```

3

```
|Password
```

```
Git: https://aspina@git.spina.me (Press 'Enter' to confirm or  
'Escape' to cancel)
```

4

```
✓ Changes committed and pushed. All done!
```

If-Statement nesting

You can nest multiple if-statements within each other.

```
int x = 8;

if (x < 10) {
    if (x < 5) {
        printf("X is less than 5\n");
    } else {
        printf("X is between 5 and 10\n");
    }
} else {
    printf("X is between greater than 15\n");
}
```

Exercise

Write a program (**fizz_buzz.c**) that given a variable **num**.

- If **num** is divisible by 2 it prints "Fizz".
- If **num** is divisible by 3 it prints "Buzz".
- If **num** is divisible by both 2 and 3 it prints "FizzBuzz".

```
Insert a number: 20  
Fizz
```

```
Insert a number: 9  
Buzz
```

```
Insert a number: 6  
FizzBuzz
```

To execute: `make fizz_buzz.run`

Exercise Solution 1

```
// Implementation:  
if ((num % 2 == 0) && (num % 3 == 0)) {  
    printf("FizzBuzz\n");  
} else if (num % 2 == 0) {  
    printf("Fizz\n");  
} else if (num % 3 == 0) {  
    printf("Buzz\n");  
}
```

Exercise Solution 2

```
bool is_div_2 = num % 2 == 0;
bool is_div_3 = num % 3 == 0;

if (is_div_2 && is_div_3) {
    printf("FizzBuzz\n");
} else if (is_div_2) {
    printf("Fizz\n");
} else if (is_div_3) {
    printf("Buzz\n");
}
```

Exercise Solution 3

```
// Implementation:  
if (num % 2 == 0) {  
    printf("Fizz");  
}  
  
if (num % 3 == 0) {  
    printf("Buzz");  
}  
  
printf("\n");
```


Save remotely your Changes

1

```
make save
```

2

```
week04 git:(master) X make save
```

3

```
|Password
```

```
Git: https://aspina@git.spina.me (Press 'Enter' to confirm or  
'Escape' to cancel)
```

4

```
✓ Changes committed and pushed. All done!
```

While-Loops

Repeat parts of
your code!

To execute:

make loop.run

```
int num;
printf("Input a number greater than 100: ");
scanf("%d", &num);

while (num <= 100) {
    printf("Wrong number, try again: ");
    scanf("%d", &num);
}

printf("Well done!\n");
```

While-Loops

Anatomy of a while-loop:

1. Uses the **while** keyword
2. The condition is between brackets:
()
3. The body is between curly brackets:
{ }

```
int x = 5;

while (x < 10) {
    x += 1;
    printf("Incr\n");
}
```

Exercise

Write a program (**hello.c**) that prints “Hello” five times.

Hint (1): use `x += 1` to increment variables

Hint (2): if the program is stuck use **Ctrl + C** from terminal

To execute:

```
make hello.run
```

Exercise - Solution

```
int x = 0;

while (x < 5)
{
    x += 1;
    printf("Hello %d\n", x);
}
```

Save remotely your Changes

1

`make save`

2

`week04 git:(master) X make save`

3

`|Password``Git: https://aspina@git.spina.me (Press 'Enter' to confirm or
'Escape' to cancel)`

4

`✓ Changes committed and pushed. All done!`

For-Loops

Repeat a **specific** amount of times!

```
int x;

for (x = 1; x <= 5; x++) {
    printf("Hello %d\n", x);
}
```

```
int x = 0;

while (x < 5) {
    x += 1;
    printf("Hello %d\n", x);
}
```

To execute:

```
make for_loop.run
```

Save remotely your Changes

1

`make save`

2

`week04 git:(master) X make save`

3

`|Password``Git: https://aspina@git.spina.me (Press 'Enter' to confirm or
'Escape' to cancel)`

4

`✓ Changes committed and pushed. All done!`

For-Loops

Anatomy of a for-loop:

1. Uses the **for** keyword
2. The condition is between brackets:
(init; condition; iter)
3. The body is between curly brackets:
{ }

```
int x;

for (x = 1; x <= 5; x++)
{
    printf("Hi %d\n", x);
}
```

Exercise

Write a program (**table.c**) that prints the first five multiples of 6:

6
12
18
24
30

or

6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30

To execute:

make table.run

Exercise - Solution

```
int x;

for (x = 1; x <= 5; x++)
{
    int product = x * 6;
    printf("6 x %d = %d\n", x, product);
}
```

Save remotely your Changes

1

`make save`

2

`week04 git:(master) X make save`

3

`|Password``Git: https://aspina@git.spina.me (Press 'Enter' to confirm or
'Escape' to cancel)`

4

`✓ Changes committed and pushed. All done!`

End of Class

See you all next week!