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	
5	Functions and Scope	
6	Arrays	
10	Advanced String Usage	DEC 19
11	Custom Data Types	JAN 9
7	Introduction to Contiki-NG and nRF52840	
8	Sensing and Actuating with Contiki-NG	
9	Timers and Concurrency	DEC 12
12	Basic Communication and Networking	JAN 16
13	Introduction to RPL and Network Routing	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

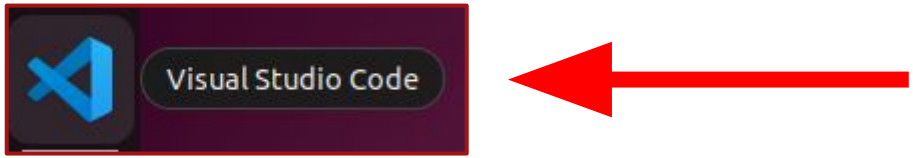
1. Turn on your Laptops
2. Login to Windows using “User”
3. Open the **Virtual Box** program
4. Add a new Virtual Machine (**Ctrl + A**)
5. Open the **VirtualBox** folder ⚠ (**NOT** the .VirtualBox)
6. Select the **nRF52840LAB** file
7. Click **Start**

Prepare the Coding Environment

1. Start the Virtual Machine **nRF52840LAB**
2. Log-in using credentials:

   Username: **ubuntu**

   Password: **ubuntu**
3. Open **Visual Studio Code** (use the App bar on the left)



Prepare the Coding Environment

4. From the Terminal:

```
make setup
```

5. 

6. 

If you see **any (yellow) errors** input the credentials again

7. Open the **`week10`** folder in the terminal

   Right click on the left + "**Open in Integrated terminal**"

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: 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-Statement chaining

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

What is the difference between these two snippets of code?

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

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

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

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

Recap: While-Loops

Repeat parts of your code!

```c
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");
```

Recap: For-Loops

Repeat a **specific** amount of times!

```
int x;

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

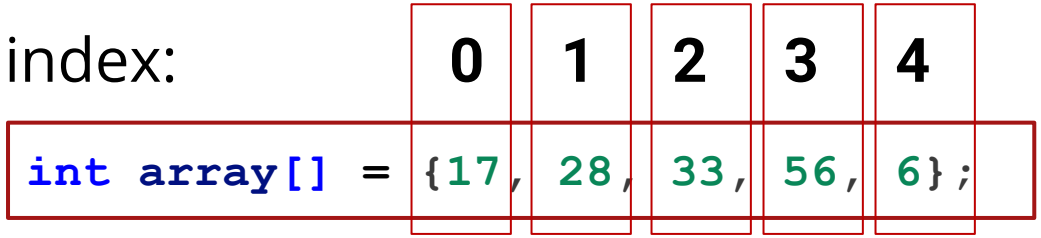
```
int x = 0;

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

Recap: Array Elements

Modifiable containers for data.

To access array elements you can use the **[index]** operator.

NOTE: List indices start from **0**




```
printf("%d\n", array[0]);
```

```
printf("%d\n", array[3]);
```

Recap: Assigning Array Elements

To assign array elements you can use the **[index]** operator on the left-hand-side of a statement (like a variable)

```
int array[] = {17, 28, 33, 56, 6};
array[3] = 100;
array[2] = -7;
```

```
printf("%d\n", array[0]);
```

```
printf("%d\n", array[3]);
```

Recap: Functions

Functions are custom snippets of reusable code:

1. If the **return type** is **void** the function does NOT return.
2. If the **return type** is NOT void, it MUST use **return**.

1

```
// Function to print a number
void print_num(int num) {
    printf("%d\n", num);
}
```

2

```
// Function to add two numbers
int add(int num1, int num2) {
    return num1 + num2;
}
```

Recap: Calling Functions

Functions can be called any number of times:

```
// Function to add two numbers
int add(int num1, int num2) {
    return num1 + num2;
}
```

```
int x = add(4, 100);
int y = add(60, 30);
int z = add(x, y);
```

The String Library

```c
/* Calculates the length of a given string  */
size_t strlen(char* str);

/* Compares strings, returns 0 when equal, 1 (or -1) otherwise  */
int strcmp(char* first_str, char* second_str);

/* Append a copy of the src string to the end of dest string */
int strcat(char* dest, char* src);
```

Exercise

Implement all functions inside (**`simple.c`**).

Function descriptions are the specification

To run automated tests: `make simple.test`

Exercise Solution

```c
size_t sum_of_string_lengths(char *str1, char *str2)
{
    return strlen(str1) + strlen(str2);
}
```

Exercise Solution

```c
bool are_three_strings_equal(char *str1, char *str2, char *str3)
{
    return (strcmp(str1, str2) == 0) && (strcmp(str2, str3) == 0);
}
```

Exercise Solution

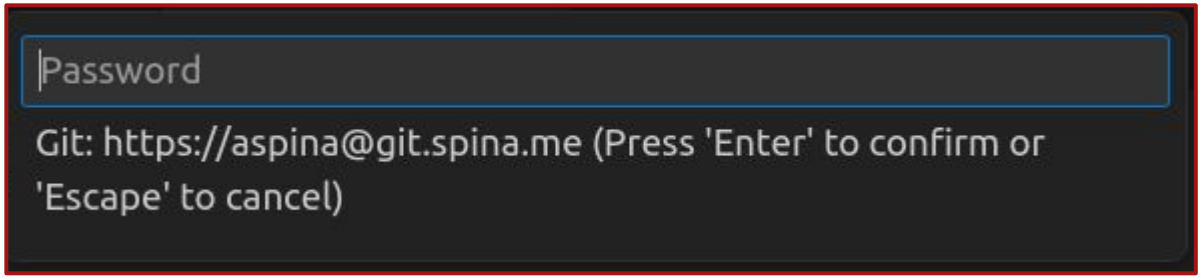
```
char *concatenate_three_strings(char *dest, char *src1, char
*src2)
{
    strcat(dest, src1);
    strcat(dest, src2);
    return dest;
}
```

Exercise Solution

```
char *find_longest_string(char *strings[], size_t num_strings) {
    size_t max_length = 0;
    char *longest_string = NULL;

    for (size_t i = 0; i < num_strings; i++) {
        size_t length = strlen(strings[i]);
        if (length > max_length) {
            max_length = length;
            longest_string = strings[i];
        }
    }
    return longest_string;
}
```

Save remotely your Changes

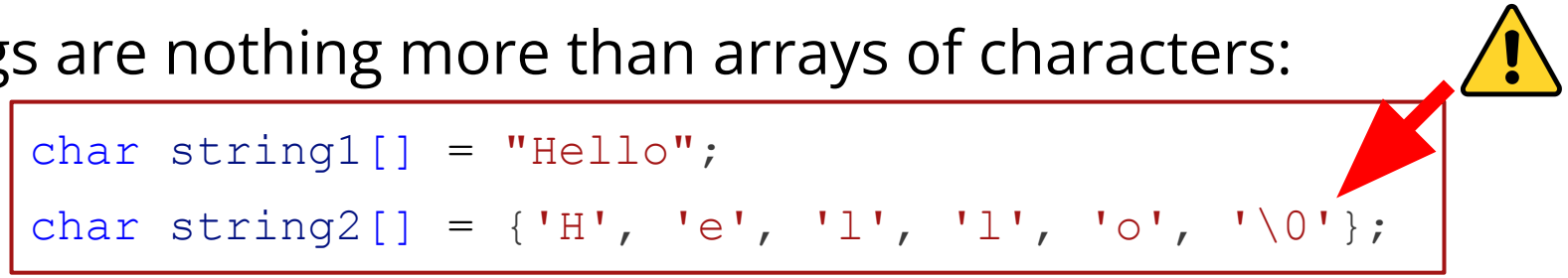
1. `make save`

2. 

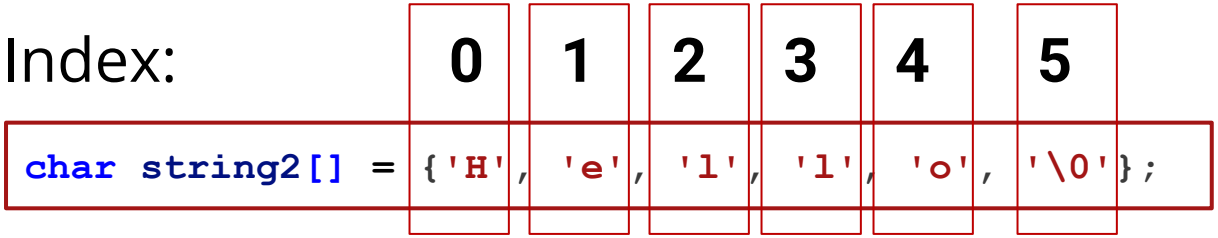
3. 

Strings as Arrays

Strings are nothing more than arrays of characters:



```
char string1[] = "Hello";
char string2[] = {'H', 'e', 'l', 'l', 'o', '\0'};
```




```
printf("%c\n", string2[0]);
```

```
printf("%c\n", string1[3]);
```

More String Library Functions

```c
/* Convert the character to lower-case */
char tolower(char character);

/* Convert the character to upper-case */
char toupper(char character);
```

Exercise

Implement all functions inside (**`advanced.c`**).

Function descriptions are the specification

To run automated tests: `make advanced.test`

Exercise Solution

```c
size_t count_char_occurrences(char *str, char character)
{
    size_t count = 0;
    for (size_t i = 0; str[i]; i++)
    {
        if (str[i] == character)
            count++;
    }
    return count;
}
```

Exercise Solution

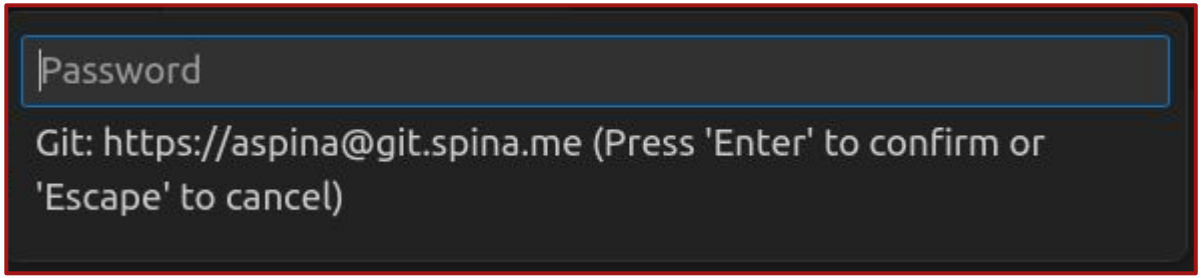
```
char *convert_to_lowercase(char *str)
{
    for (int i = 0; str[i]; i++)
    {
        str[i] = tolower(str[i]);
    }
    return str;
}
```

Exercise Solution

```c
void reverse_string(char *str)
{
    size_t length = strlen(str);
    for (size_t i = 0, j = length - 1; i < j; i++, j--)
    {
        char temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}
```

Save remotely your Changes

1. `make save`

2. 

3. 

Trivia Time!


Join at:
ahaslides.com/
CI5KK
AhaSlides
Christmas Trivia Quiz
ahaslides.com/CI5KK
Menu
0/7

End of Class

See you all next week!