IEEE RAS
ROBOTICS & AUTOMATION SOCIETY
Zagazig student chapter

TASK 1

Embedded Systems Track

DATE :

22/3/2025

PRESENTED BY :

MOHAMED KADRY

Table Of Content

1 **Declaring and Initializing Variables in C.**

Variable Declaration

In C, a variable must be declared before it can be used.
The syntax is:

```
data_type variable_name;
```

Example:

```
int age;      // Integer variable
float salary;  // Floating-point variable
char grade;    // Character variable
```

Variable Initialization

Initialization assigns an initial value to a variable at the time of declaration.
The syntax is:

```
data_type variable_name = value;
```

Example:

```
int age = 25;                  // Integer initialized to 25
float salary = 5000.50; // Float initialized to 5000.50
char grade = 'A';           // Char initialized to 'A'
```

C keywords:

they are a special word for the preprocessor,
auto - double - int - float - struct - break - continue - if - else - switch - long - case - enum - register - typedef - char - extern - do - return - while - for - union - void - signed - static - default - goto - sizeof - volatile - const - restrict - short.

1 **Declaring and Initializing Variables in C.**

Data Types:

-each data type determines the size of the variable and the range is determined using the modifiers (*signed* and *unsigned*), but *short, long, and long long* determine the size only, and
note that *Short x;* = *Short int x;* by default.
The range of the data types is calculated through this relationship:
$-2^{n-1} : 2^{n-1} - 1$ if it's signed
but if it's unsigned:
$0 : 2^{n} - 1$

for example

if we have signed *char x = 135;* the signed range here is *-128:127* so *135* is out of the range, we can get the steps by doing *135 - 127 = 8* steps, then we move 8 steps From *135* till we reach *-121*, or we can say directly -128 + steps - 1 = -128 + 8 - 1 = -121. Otherwise, we can convert the *135* to binary form *10000111* and we take only the first 8 bits (1 Byte) *10000111* which equals *-121*.

Type	Size	Value Range (min)
signed char int8_t	At least 8-bits	$[-2^7, +2^7 - 1]$
unsigned char uint8_t	At least 8-bits	$[0, +2^8 - 1]$
signed short int int16_t	At least 16-bits	$[-2^{15}, +2^{15} - 1]$
unsigned short int uint16_t	At least 16-bits	$[0, +2^{16} - 1]$
signed long int int32_t	At least 32-bits	$[-2^{15}, +2^{15} - 1]$
unsigned long int uint32_t	At least 32-bits	$[0, +2^{32} - 1]$

-C data types are primitive data types (int, float.. etc.), derived data types (array, pointer, struct.. etc.), and user-defined data types (typedef and enum).

2 Performing Arithmetic Operations with Numeric Variables in C

Arithmetic operators : [() * % / + -] in priority.

Boolean expressions: operators > < <= >= == != , they return True or False. C doesn't have a boolean data type but in c99 there's a boolean data type in <stdbool.h> that takes two values true (1) or false(0). We shouldn't use floating-point numbers here.

Logical expressions: operators && AND - || OR - ! NOT, they return True or False

Bitwise expressions: Bitwise operators = Binary operators which convert the operands to the binary form
(<<, >>, |, &, ~, ^) are used with unsigned integers.

Complement operator (~) sets 1 if the bit is zero and vice versa.
Xor operator (^) sets 1 if the two bits are different.
And and Or operators (& and |) are well-known.
Left-Shift operator (<<) shifts the bits of the 1st operand left by the numbers of bits specified by the second operand and fills from the right with 0 bits
(e.g. 1 << 2 = 1 * 22 = 1 * 2 * 2 = 4 or 0000 0001 << 2 = 0000 0100).
Right-Shift operator (>>) shifts the bits of the 1st operand right by the number of bits specified by the second operand and the method of filling from the left is machine-dependent when the left operand is negative. The right operand (x) in 1 << x or 1 >> x mustn't be negative and shouldn't be larger than the number of bits in which the left operand is stored.

Comma Operator vs. Comma Separator:
The comma (,) operator returns the rightmost operands in the expression and it simply evaluates the rest of the operands. It has the least precedence among all other operators, for example, int a = (1, 2, 3); a = 3, but if int a = 1, 2, 3; it will lead to syntax errors because the compiler looks at the last statements as int 2; int 3; so identifiers here wrong, a = 3, 4, 5 here a = 3 because the assignment operator here is higher than the comma (,) in the precedence. Here the comma (,) is a separator

2 Performing Arithmetic Operations with Numeric Variables in C

Post/Pre Operators:

When using the post-increment with a variable and assigning it in the same variable like that `int i = 3; i = i++;` i will still equal 3 and the updated value will be discarded because it's not used in the code, especially after the assigning process.

Precedence, Associativity, and Order of Evaluation:

OPERATOR	TYPE	ASSOCIAVITY
() [] . ->		left-to-right
++ -- + - ! ~ (type) * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
|	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
||	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operator	right-to-left
,	Comma	left-to-right

3 Using Conditional Statements in C.

-We have **3 control structures in C**, sequence structure, selection structure, and repetition structure.

The 'If' selection statement

It acts if the condition is true, and skips the action if the condition is false.
The 'If-Else' selection statement acts if the condition is true, and performs a different action if the condition is false.
Note that we can use scanf() or printf() inside some expressions of if-else, while, or for statements as they return 0 or 1 related to their type of inputs.

```
if (condition) {
    // Executes if condition is true
} else {
    // Executes if condition is false
}
```

The 'Switch' selection

```
switch (expression) {
    case value1:
        // Code block
        break;
    case value2:
        // Code block
        break;
    default:
        // Code block if no case matches
}
```

performs one of many different actions depending on the value of expressions. **Switch(variable/controlling expression) {case 1.....}** The **variable must be an integral variable** which means int, char, and long but not float. The switch **is better than (If-Else) in speed at runtime**, because it jumps directly to the right case based on the look-up table, conversely, in (If-Else) it passes through all statements, but the labels in the switch statement are restricted with only integral constant, not variables and these labels are the result of the switch expressions which must be integral constants (char, int, enum) only, **so (If-Else) has the higher capability.**

The *break* is necessary in any case to exit the switch except if we want to run a certain case always and it's optional in the default statement, if there is no break statement in the switch statement, all the case statements will follow execution until encountering a break statement or the end of the switch if the first case is true.
Any statement that is outside the scope of the labels of the switch-case statements will not be executed.

4 Iterating with Loops in C.

While, Do-While, and For loops

```
for (initialization; condition; increment) {
    // Code block
}
```

```
while (condition) {
    // Code block
}
```

```
do {
    // Code block
} while (condition);
```

While repetition statement also called an iteration statement, which has a specific action that is repeated while some condition is true.

The do-while repetition statement is similar to the while statement, but the difference is the body is executed one time if the condition of the while is false.

For loop, it's deterministic and conditional where for (initialization statement; test expression (condition); update statement(increment/decrement) {body}. The initialization statement here occurs one time only. The conditions here shouldn't be floating-point numbers or control variables otherwise it will lead to logical errors.

5 Using Constants in C.

The const keyword is used to create read-only variables that cannot be modified after initialization
Constant can be any of the basic data types like : Integer - Floating - Character - String literals.

Syntax

```
const data_type VARIABLE_NAME = value;
```