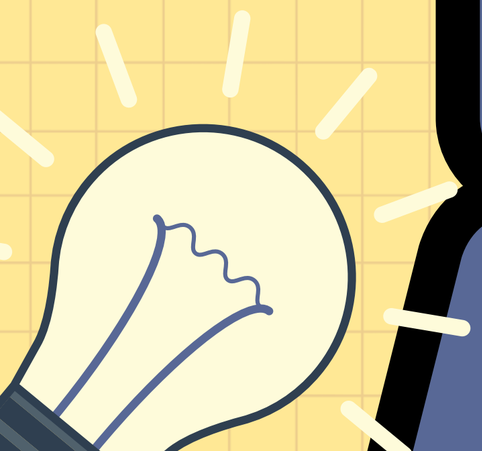
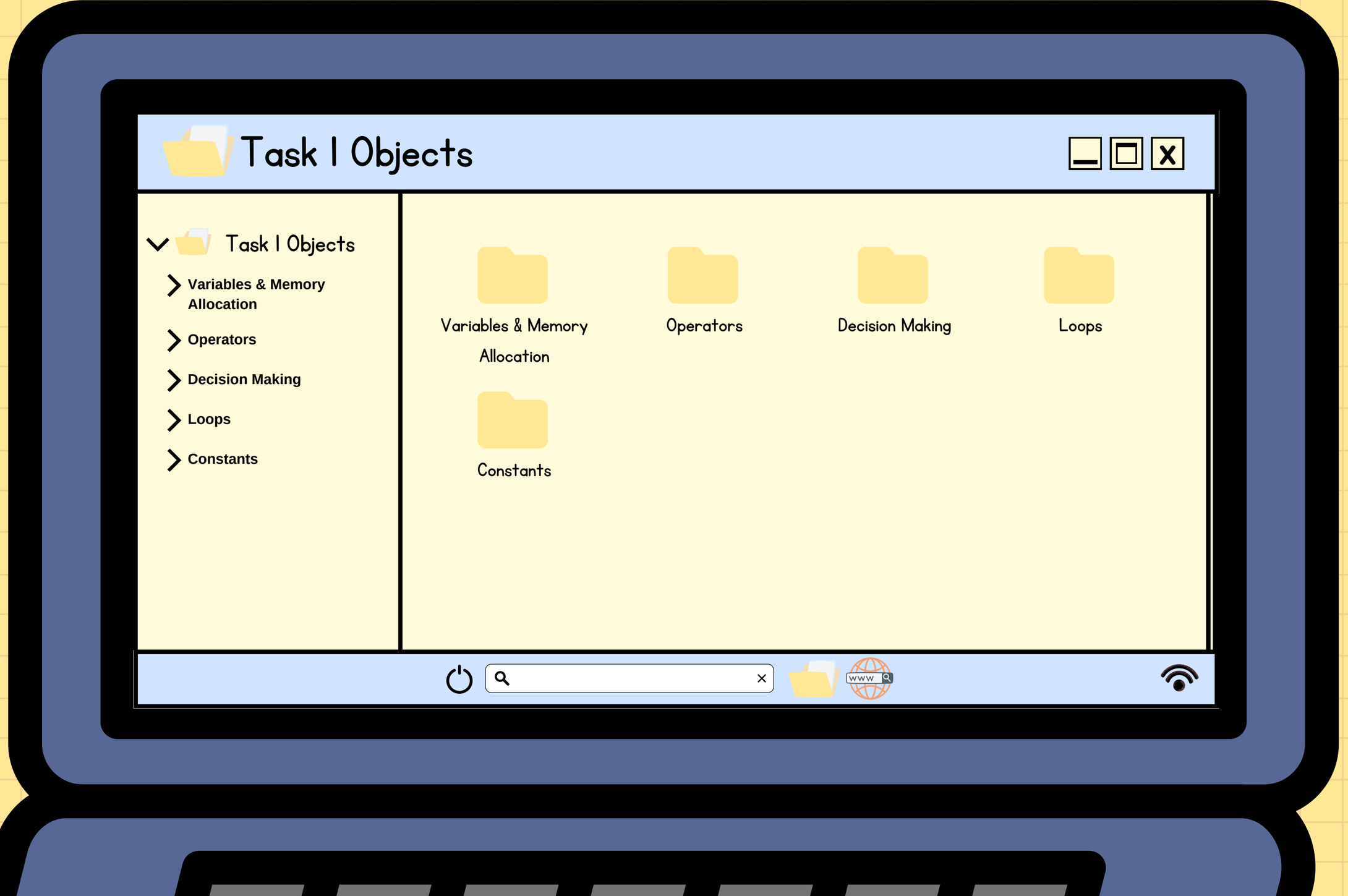
Embedded Systems

Task 1

by Mohammed Elahmady



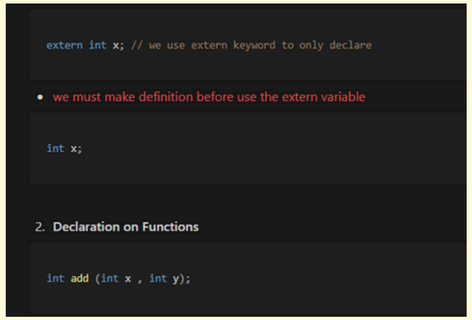

Task 1 Objects
Task 1 Objects
Variables & Memory Allocation
Operators
Decision Making
Loops
Constants
Variables & Memory Allocation
Operators
Decision Making
Loops
Constants
www

Variables & Memory Allocation

Declaration :

- The declaration has no space in memory and cannot store a value.
- The compiler declares that a variable or function exists, but it cannot be used until it has been defined.
- If there is an attempt to access a variable that has only been declared or call a function that has only been declared without a definition, it results in a type of error called a linker error.

1. **Declaration on Variables**
2. **Declaration on Functions**

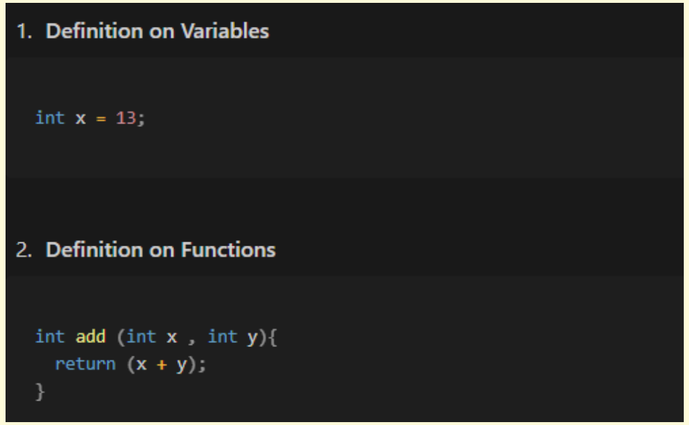


Variables & Memory Allocation

Definition :

- The definition allocates space in memory, allowing you to access it and assign a value to it.
- When a definition is provided, the compiler fully defines the variable, allowing you to assign a value to it, or the function, enabling you to call it.

1. **Definition on Variables**
2. **Definition on Functions**



Operators

Operators :

- an operator is a symbol that tells the compiler to perform specific mathematical or logical function.

1. Arithmetic Operators :

- Arithmetic operators in C are used to perform basic mathematical operations on numerical values. Here are the main arithmetic operators:

Arithmetic Operators :

```c
+  // => addition
-  // => substraction
*  // => multiplication
/  // => division
%  // => Reminder
++ //  => increment
-- // => decrement

//Examples
int A = 20 , B = 10;
int sum = 0;
sum = A+B; // sum = 30
sum = A-B; // sum = 10
sum = A*B; // sum = 200
sum = A/B; // sum = 2
sum = A%B; // sum = 0

sum = ++A; // sum = 21 - A = 21 // Pre increment
sum = --A; // sum = 20 - A = 20 // Pre decrement
sum = A++; // sum = 20 - A = 21 // post increment
sum = A--; // sum = 21 - A = 20 // post decrement
```

Operators

2. Relational Operators :

- Its operands are usually integers, floats, or even characters, and the result is either 0 or 1.

```
==  // => is the right equal the left
!=  // => is the right not equal the left
>   // => is the right greater than the left
<   // => is the right less than the left
>=  // => is the right greater than or equal the left
<=  // => is the right less than or equal the left

//Examples
int A = 50, B = 23;
int sum = 0;
sum = A==B; // sum = 0
sum = A!=B; // sum = 1
sum = A>B;  // sum = 1
sum = A<B;  // sum = 0
sum = A>=B; // sum = 1
sum = A<=B; // sum = 0
```

Operators

3. Logical Operators :

- Its operands always have values of 0 or 1, and the result is always 0 or 1.

```
&&  // Logical AND => the 2 conditions must be true
||  // Logical OR => at least 1 condition must be true
!   // Logical NOT => not of the operand 0 => 1 , 1 => 0

//Examples
int A = 1, B = 0;
int sum = 1;
sum = A&&B; // sum = 0
sum = A||B; // sum = 1
sum = !A;   // sum = 0
```

Operators

4. Bitwise Operators :

- Each bit interacts with its corresponding bit in the other operand.

```
&  // => Bitwise AND
|  // => Bitwise OR
^  // => Bitwise XOR
~  // => Bitwise NOT
<<  // => Bitwise Left Shift
>> //  => Bitwise Right Shift

short A = 60;   // A   = 0011 1100
short B = 13;   // B   = 0000 1101
short sum = 0;  // sum = 0000 0000
sum = A & B;    // sum = 0000 1100 => 12
sum = A | B;    // sum = 0011 1101 => 61
sum = A ^ B;    // sum = 0011 0001 => 49
sum = ~A   ;    // sum = 1100 0011 => 195
sum = A << 1;   // sum = 0111 1000 => 120
sum = A >> 1;   // sum = 0001 1110 => 30
```

Operators

5. Assignment Operators :

- Assignment operators are used to assign values to variables. The basic assignment operator is =. There are also compound assignment operators that perform an operation and assign the result in one step.

```
=    // => assign
+=   // => add and assign
-=   // => sub and assign
*=   // => multiply and assign
/=   // => divide and assign
%=   // => remind and assign
>>=  // => right shift and assign
<<=  // => left shift and assign
&=   // => bitwise and and assign
|=   // => bitwise or and assign
^=   // => bitwise xor and assign

//Examples
int A = 20 , B = 2;
A += B;                          // A = 22
A = 20;

A -= B;                          // A = 18
A = 20;

A *= B;                          // A = 40
A = 20;

A /= B;                          // A = 10
A = 20;
```

Operators

6. Conditional Operators :

- The conditional operator in C is a ternary operator (?:) that provides a shorthand way to write simple if-else statements. It has the following syntax:

7. Other Operators :

Conditional Operator :

```
Var = EXP1?EXP2:EXP3;
if=> EXP1 = true  => Var = EXP2;
if=> EXP1 = false => Var = EXP3;

//Example
int A = 23, B=20;
int sum = (A>B)?15:10;
```

Others Operators :

```
sizeof(data type / variable) => size of data type or variable
&  => a refrence     => return address of variable or array or pointer
*  => a refrence     => return value of address that assigned in a pointer
.  => dot operator   => struct , enum , union
-> => arrow operator => struct , enum , union
```

Decision Making

Decision Making :

1. **if Statement :**
- if statement is the simplest way to make a decision making
- we use if statement to control the program flow based on some condition

```c
//Syntax

if(condition){
    /*Condition Code*/
}
```

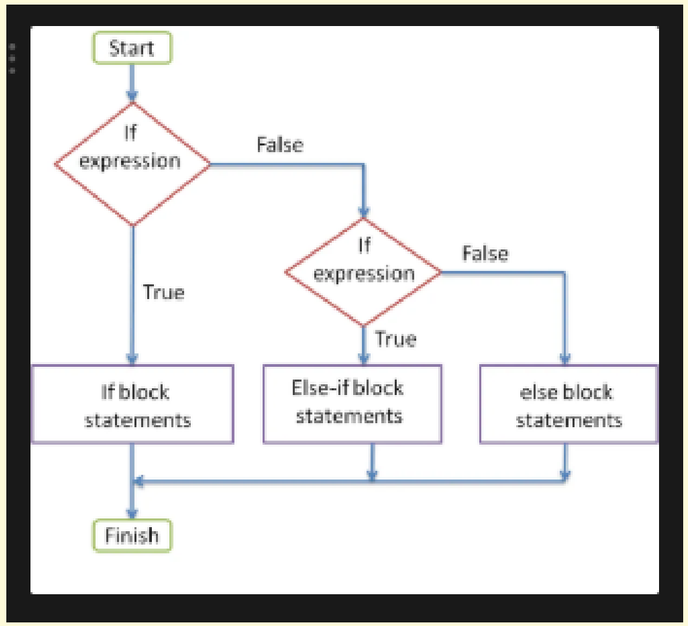
Decision Making

Decision Making :

2. **else if , else Statements :**

```
//Syntax

if(condition1){
    /*Condition1 Code*/
}
else if(condition2){
    /*Condition2 Code*/
}
else{
    /*else Code*/
}
```



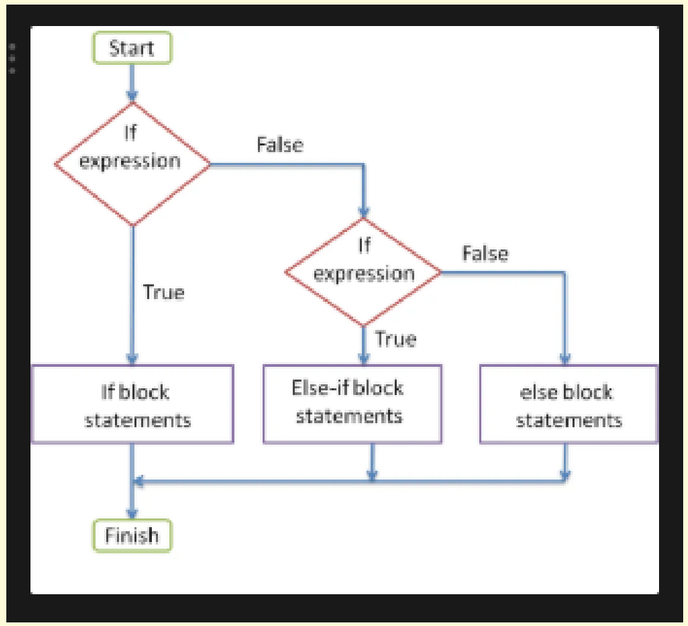
Decision Making

Decision Making :

3. Switch Statement :

```
//Syntax

if(condition1){
    /*Condition1 Code*/
}
else if(condition2){
    /*Condition2 Code*/
}
else{
    /*else Code*/
}
```



Decision Making

Decision Making :

3. **Switch Statement :**

- C Switch Statement is used when you have multiple possibilities for the if statement .

```
//Syntax

switch(Variable){
    case():
    /*Case Code*/
    break;
    case():
    /*Case Code*/
    break;
    case():
    /*Case Code*/
    break;
    default:
    /*default Code*/
    break;
}
```

Loops

Loops :

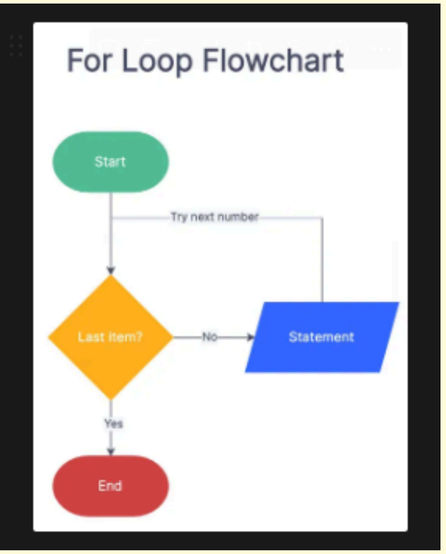
1. **For Loop :**

```
//Syntax

for(start;end;update_step){
    /*Loop Body*/
}
```

- to write infinite loop we have 2 methods :

```
for(;;);
for(start;end;){}
```



Loops

Loops :

2. While Loop :

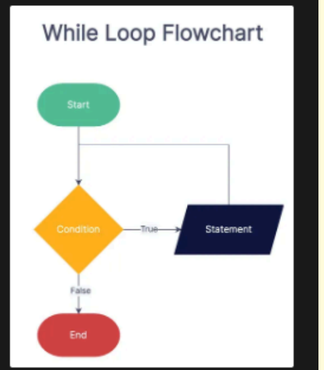
- the while statement execute statement or block of statements until the given condition is true.

```
//Syntax

while(condition){
    /*Loop Body*/
}
```

- to write infinite loop :

```
while(1);
```



Loops

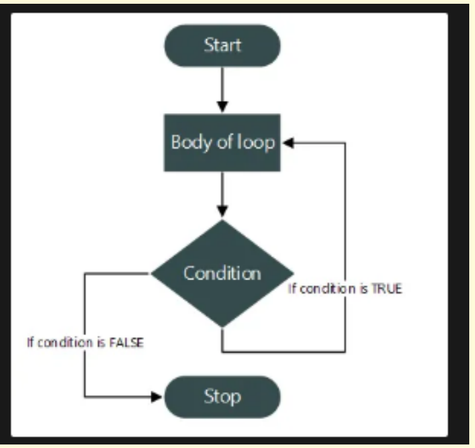
Loops :

3. **do while Loop :**

- the body of do while loop is executed once before checking the while condition.

```
//Syntax

do{
    /*Loop Body*/
}while(condition);
```



Constants

Constants :

3. **do while Loop :**

- Constant refers to fixed values that the program may not alter during execution.
- Constant can be any of the basic data types like :
- Integer - Floating - Character - String literals

Example :

- Example

```
Const float pi = 3.14;  // Const global variable "Read only" can't be changed [Direct - I

int main(){
    const int num = 55; // Const global variable "Read only" can't be changed [Direct]
    retutn 0;
}
```

THANK YOU

Head : Tasnem Sabry

Vice : Ahmed Yasser