**Introduction To programming Using C language**

**Lesson 1 : output – input functions**

* Function Definition
* C Program Structure
* Variable Definitions
* Complete program structure
* The printf() function ( accepts a wide variety of “Format Codes” %d ,%f , …..)
* Field-Width Specifiers
* Escape Sequences ( \n , \t , …. )
* scanf() Function
* The Address Operator (&) & called ampersand
* scanf() - Format Specifiers
* C Keywords – Reserved Words
* program to add two numbers

**Lesson 2 : Variable Definitions**

* identifiers
* Explicit – implicit declaration
* Variable Definitions
* Variable types char, int, float , double
* preferred place to Declare Variable ( in the beginning of main body )
* Variable type modifier ( signed, unsigned, short, long )
* Data representation 1 Byte=8 Bits
* C Basic Data Types
* sizeof() Operator
* Program to determine data types size in byte using sizeof operator
* Find range of data types in C
* Find range of data types using C library (In C programming minimum and maximum constants are defined under two header files – limits.h and float.h)
* Local – Global variables

**Lesson -3 : assignment operator-literals- comments**

* Assignment Operator =
* Initializing Variables
* Literals in C
* Integer Literals 1-Prefixes (Examples: X=56; X=056; X=0x46A; X=ob10101; ) 2-Suffixes (Examples int i=12l;)
* Defining Constants #define 🡪 text replacement , const
* Renaming Data Types with typedef Ex: typedf unsigned char Byte; Byte a;
* Comments /\* \*/ , //

**Lesson -4 : Arithmetic Operators- Expressions**

* Operators
* Arithmetic Operators
* Division and Modulus ( Modulus=remainder )
* The Order of Precedence 🡺 \* , / , % , + , -
* Parentheses ()

**Lesson -5 : Assignment operator**

* Assignment operator
* Multiple Assignments a = b= c = d = o = 100;
* Compound Assignment total = total + number; ➔ total += number;

**Lesson -6 : Increment and Decrement Operators**

* Increment Operator ++
* Decrement Operator --

**Lesson -7 : Relational Operators**

* Relational operators < , > , <= , >= , == , !=

**Lesson -8 : Logical Operators**

* Logical Operators && , || , !
* Compound condition

**Lesson -9 : The ( ?:) Conditional operator**

* Conditional operator conditional\_expression? expression1: expression2;

**Lesson -10 : Bitwise Operators**

* Bitwise Operators & , | , ^ , ~
* Compound Bitwise Operators &= , |= , ~=
* The Bitwise Shift Operators << , >>

**Lesson -11 operator precedence – Associativity**

* Operator precedence ) EX: the multiplication operator has a higher precedence than the addition operator (
* Operator associativity ) If multiple operators with the same precedence )

**Lesson -12: Type Conversion and Casting**

* Type Conversion
* Promotion – Demotion (promotion char < int < long < float < double , Demotion has table)
* Typecasting ex : answer = (int) fltnum;

**Lesson -13 : Branching**

* If Statement if (expression) {statement;}
* Multiple Statements With if if (expression) { statement; statement; statement; }
* if-else Statement if (expression) {statement; else statement; }
* Nested If
* The Else-If Construct

**Lesson -14 : The switch Statement**

* switch case statement

**Lesson -15 : for statement**

* For statement for (initializationStatement; testExpression; updateStatement) { // statements }
* Multiple Statements in Loops
* Nested for
* Multiplication table
* The Break And Continue Statements

**Lesson -16 : while Statement**

* While statement while (condition) { // code block to be executed}

**Lesson -17 : The Do While Loop**

* Do While Loop do { // the body of the loop} while (testExpression);
* Fibonacci Series

**Exercises :**

**Lesson -18 : Functions**

* The Function Definition
* function prototype
* Calling The Function
* Advantages Of Prototyping
* Local Variables
* The Return Statement
* Pass Data To A Function
* call by Value

**Lesson -19 : variable Scope or availability**

* Scope –availability ( File or global Scope , Function Scope local scope , Function Prototype Scope , Block Scope )

**Lesson -20 - variable Storage Classes or Lifetime**

* Storage Classes
* Lifetime
* Automatic Variables ( auto )
* Static Variables
* external Variables
* Register Variables

**Lesson -21 : recursion**

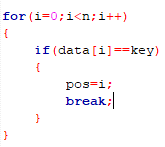
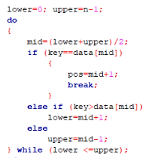
* Recursion with Return Value
* Recursion without Return Value
* Two level Recursion ( I will study this point in data structures (tree recursion ) , there are 5 types of recursion I will study it in data structures )
* How to think recursion

**Lesson -22 : Arrays**

* Needs of Arrays
* Array Definition int day1 , day2 , day3 , day4 , day5 , day6 , day7 ; ======➔ int day[7];
* Array Representation
* Reading In An Unknown Number Of Elements
* Bounds Checking
* Initializing Arrays
* Access Arrays elements using Array size sizeof(array) / sizeof(array[0])
* Arrays As Arguments
* Addresses Of Things Versus Things the first element of the array, it will have the same address as the array itself array=&array[0]
* Function call by value or by reference

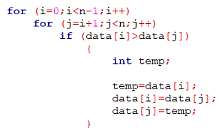
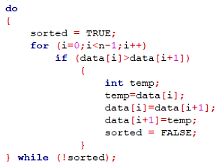
**Lesson -23 : Arrays Operations**

* Printing Array
* Reading Array
* Insert Element to Array
* Delete Element From Array

**Lesson -24 : Search a specific value in Array**

* Linear Search
* Binary Search

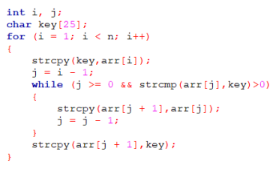
**Lesson -25 : Sorting the Arrays Elements**

* ****Selection sort
* Bubble sort

**Lesson -26 : Two dimension Arrays**

* Two dimensional Arrays int arr[row][col];
* Initializing Two-Dimensional Arrays
* store the elements of two dim Array
* Magic square as two dim
* Magic square checks
* Sparse Matrix 1-Triplet Representation (Array Representation)

**Lesson -27 : strings**

* String data type char String\_Variable\_name [ SIZE ] ;
* String variable with laterals initialization
* String variables
* The String I/O Functions gets() And puts()
* String as function parameter ( palindrome example )
* String functions <string.h>
* String functions Examples ( strlen(),strcmp(),strcpy,…. )
* Sorting array of strings
* insertion sort

**Lesson -28 : pointers**

* address of the variable - &
* Pointers int \*ptr; int \* ptr;
* Pointers data type (4 bytes)
* Dereferencing Pointer operator (\*)
* void Pointer
* pointer arithmetic
* Incrementing Pointer new value = current adress + size\_of(data type)
* C difference between \*ptr++ & ++\*ptr
* Double Pointer (Pointer to Pointer) int \*\*ptr;
* NULL Pointer int \*ptr = NULL;
* Difference between char \*a and char a[]
* Constant Pointers ( cannot change the address of the variable ) int \* const ptr=&x;
* Pointer to Constant ( cannot change the value of variable ) const int \* ptr=&x;
* Pointer to function return type (\*ptr\_name) (type1, type2…);
* Pointer to array element x [ i ] = \* ( x + i )= \* ( i + x)=i[x]
* Pointer to Character = string char \* name = “youssef"; printf(“%s\n”,name);
* Pointer to array of string char \*arr[2] = {"Java",“Python"}; char \*(\*ptr)[2] = &arr;
* Pointer usages

1. passing address to function ( call by references )
2. Return more than one value from a function
3. Pass arrays and strings more conveniently return array as pointer
4. Manipulate arrays more efficient by moving pointers
5. DirectAddress memory location

**Lesson -29 : structures**

* Structures
* Declaring A Structure Type struct tag\_name { type member1; type member2; };
* Defining Structure Variables struct tag\_name Variable;
* Accessing Structure Members use the dot syntax (.)
* Combining Declarations
* Initializing Structures
* Assignment Statements with Structures
* Pointer to Structures ( EX: struct person \* ptr; ptr = & person1; ptr->code =230; )
* Passing Structures To Functions
* return Structure from Function
* Arrays Of Structures
* Dynamic Memory Allocation ptr = (castType\*) malloc(size); free(ptr);
* Arrays Of pointers to Structure
* Size of Structure-struct padding
* Pointers And Structures “The Linked List”
* linked list in Queue Style (FIFO) (First In First Out)
* linked list in Stack Style (LIFO) (Last In First Out)
* Union in C Language
* Defining struct Bit Fields in C

**Lesson -30 : File I/O**

* input and output (I/O) to a disk
* Character, String, Formatted, And Record I/O
* Text Versus Binary
* Working with files FILE\*fptr; fptr=fopen(“FileOpen”,”mode”); fclose(fptr);
* Character Input/Output getc() getche() putc()
* Trouble Opening The File
* command line arguments argc (argument count) argv (argument array of strings)
* Generalized copy file
* String (line) Input / Output fputs() fgets()
* Formatted Input / Output fprintf() fscanf() fgets()
* Number Storage in text-binary format fread() and fwrite()
* Reading and writing to a binary file
* Record Input / Output
* Writing Structures With fwrite()
* reading Structures With fread()
* Random Access fseek()
* Random Access fseek()- Example