3. Programming Language and its applications

3.1 Introduction to C Programming

C-Tokens

- A token is the smallest unit used in a C program.
- Each and every punctuation and word that you come across in a C program is token.
- **Classification and Types of Tokens in C**
  - Identifiers in C
  - Keywords in C
  - Operators in C
  - Strings in C
  - Special Characters in C
  - Constant in C

Identifiers:

- These are used to name the arrays, functions, structures, variables, etc.
- The identifiers are user-defined words in the C language.
- These can consist of lowercase letters, uppercase letters, digits, or underscores, but the starting letter should always be either an alphabet or an underscore.
- **Rules:**
  - The identifiers must not begin with a numerical digit.
  - The first character used in an identifier should be either an underscore or an alphabet. After that, any of the characters, underscores, or digits can follow it.
  - Both- the lowercase and uppercase letters are distinct in an identifier. Thus, we can safely say that an identifier is case-sensitive.
  - We cannot use an identifier for representing the keywords.
  - An identifier does not specify blank spaces or commas.
  - The maximum length of an identifier is 31 characters.
  - We must write identifiers in such a way that it is not only meaningful- but also easy to read and short.

Keywords:

- the keywords as the reserved or pre-defined words that hold their own importance.

auto	enum	const	goto
double	case	float	default
struct	register	unsigned	sizeof
int	typedef	short	volatile
break	extern	continue	if
else	char	for	do
switch	return	void	static
long	union	signed	while

Operators:

- The operators in C are the special symbols that we use for performing various functions.
  - Unary Operator: single operand; eg ++, --, (type)*, sizeof
  - Binary Operator: two operands; eg
    - Relational Operators: ==, !=,>,<,>=,>=
    - Arithmetic Operators:: -,+,/,*
    - Logical Operators: !, ||, &&
    - Shift Operators
    - Conditional Operators
    - Bitwise Operators: &,|,~,,<<,>>
    - Misc Operator: size of(), &,*,?:
    - Assignment Operator: =, +=, -+, *=, /=,%=
  - Ternary Operator: three operands. Eg: (age >= 18) ? printf("Can Vote") : printf("Cannot Vote");

String

- The strings in C always get represented in the form of an array of characters.
- a '\0' null character at the end of any string- thus, this null character represents the end of that string.
- Eg: char x[] = 'chocolate';

  char x[9] = {'c','h','o','c','o','l','a','t','e','\0'};

Special Characters in C

- () Simple brackets – We use these during function calling as well as during function declaration.
- [ ] Square brackets – The closing and opening brackets represent the multidimensional and single subscripts
- (,) Comma – We use the comma for separating more than one statement, separating the function parameters used in a function call, and for separating various variables when we print the value of multiple variables using only one printf statement.
- { } Curly braces – We use it during the closing as well as opening of any code. We also use the curly braces during the closing and opening of the loops.
- (*) Asterisk – We use this symbol for representing the pointers and we also use this symbol as a type of operator for multiplication.
- (#) Hash/preprocessor – We use it for the preprocessor directive. This processor basically denotes that the user is utilizing the header file.
- (.) Period – We use the period symbol for accessing a member of a union or a structure.
- (~) Tilde – We use this special character in the form of a destructor for free memory.

Constant in C

- Constant is basically a value of a variable that does not change throughout a program.
- we can declare a constant:
  - By using a #define pre-processor
  - By using a const keyword

Type of Constant	Example
Floating-point constant	25.7, 87.4, 13.9, etc.
Integer constant	20, 41, 94, etc.
Hexadecimal constant	0x5x, 0x3y, 0x8z, etc.
Octal constant	033, 099, 077, 011, etc.
String constant	“c++”, “.net”, “java”, etc.
Character constant	‘p’, ‘q’, ‘r’, etc.

Practice Problems on Tokens in C

1. Which of the following operators is used to concatenate two strings without space?
   a) # b) < > c) ** d) ##
2. Identify the rules that must be followed in the case of identifiers:

a. The identifiers must begin with a numerical digit.

b. The first character used in an identifier should be either an underscore or an alphabet.

c. The lowercase and uppercase letters are not distinct in an identifier.

d. An identifier does not specify blank spaces or commas.

e. The maximum length of an identifier is 31 characters.

A. a, c, and e **B.** b, d, and e **C.** a, b, and d **D.** b, c, and e

3. Which of these is not a constant used in the C language?

A. Octal Constants **B.** String Constants **C.** Integral Constants **D.** Floating Constants

- **Is there a difference between constants and literals?**

No. Both of these are the same. The syntax of a constant goes as follows: const data_type variable_name = value;

- **Do keywords have functions pre-defined for them, and can we use them for variables?**
- Yes, keywords have functions pre-defined for them, but we cannot use these to assign names to the variable. If we do so, it would connote a very different meaning, and we will experience a lot of errors.

Precedence of operators:

Highest precedence top to bottom

Type of Operator	Associativity	Category
() [] -> . ++ – –	Left to right	Postfix
– + ! ~ – – ++ (type)* & sizeof	Right to left	The Unary Operator
/ * %	Left to right	The Multiplicative Operator
– +	Left to right	The Additive Operator
>> <<	Left to right	The Shift Operator
< > >= <=	Left to right	The Relational Operator
!= ==	Left to right	The Equality Operator
&	Left to right	Bitwise AND
^	Left to right	Bitwise XOR
|	Left to right	Bitwise OR
&&	Left to right	Logical AND
||	Left to right	Logical OR
?:	Right to left	Conditional
= -= += /= *= %=>>= &= <<= |= ^=	Right to left	Assignment
,	Left to right	Comma

Practice Problems on Operators in C

1. Take a look at the following program:

```
#include <stdio.h>
int main() {
        int x;
        x = 2;
        printf( "%d\n", x );  2
        printf( "%d\n", x++ );post increment 2
        printf( "%d\n\n", x ); 3
        x = 2;
        printf( "%d\n", x ); 2
        printf( "%d\n", ++x ); pre increment; 3
        printf( "%d\n", x );  3
        return 0;
}
```

Output:
The output of the code mentioned above will produce a result as:

A. 2 2 3 2 3 3 **B.** 2 3 3 2 2 3 **C.** 3 2 3 3 3 2 **D.** 3 3 2 3 2 2

2. Take a look at the following snippet of a code:

```
int a = 10;
int b = a++%5;
```

What will be the value of a and b after we execute the code?

A. a is 10, and b is 1. **B.** a is 10, and b is 0.

C. a is 11, and b is 0. **D.** a is 11, and b is 1.

3. Take a look at the following snippet of code:

```
int a = 10;
int b = ++a%5;
```

What will be the value of a and b after we execute the code?

A. a is 10, and b is 1. **B.** a is 10, and b is 0.

C. a is 11, and b is 0. **D.** a is 11, and b is 1.

Formatted input and output

- Printf():formatted output
  - **Syntax for printf()**
  - printf (format_specifiers, info_a, info_b,……..);
  - Printf(“%d”, a);

- Scanf(): formatted input
  - **Syntax for scanf()**
  - scanf (format_specifier, &data_a, &data_b,……); // Here, & refers to the address operator
  - Scanf(“%d”, &a);

1. What will be the output of the following C code?

```
#include <stdio.h>
    int main()
    {
        int n;
        scanf("%d", n);
        printf("%d\n", n);
        return 0;
    }
```

a) Compilation error b) Undefined behavior
c) Whatever user types d) Depends on the standard

Control Statements

1. If Statements
2. Switch Statement
3. Conditional Operator Statement
4. Goto Statement
5. Loop Statements

1. The If Statements:

- Simple else or Null else
- Else if ladder
- Nested if
- If… else

- **The If… Else Statement:**

```
if (condition 1) {
Statement 1 (s1); }
Else {
Statement 2 (s2) }
```

- **The Nested If Statement:**

```
If (condition 1) {
If (condition 2) {
Statement 1 (s1); }
Else {
Statement 2 (s2)
}
}
```

- **The Else If Ladder:**

```
If (condition 1) {
Statement 1 (s1); }
Else if (condition 2) {
Statement 2 (s2); }
else if (condition 3) {
Statement 3 (s3) }
…
Else{
Statement 4 (s4)
}
```

- **The Simple Else or Null Else**

```
If (condition1)
{
Statement 1 (s1);
}
Statement 2 (s2);
```

The Switch Statements:

```
switch(expression or variable)
        {
            case val-1: // colon not semicolon
            statement-1;
            break;

            case val-2: // colon not semicolon
            statement-2;
            break;
            - - - - - - -
            - - - - - - -
            case val-n:
            statement-n;
            break;
            default:   // colon not semicolon
            statement;
        }
```

Loop:

1. while Loop: While loop executes the code until the condition is false

```
while(condition){
//code }
```

2. do – while loop: It also executes the code until condition is false. In this at least once, code is executed whether condition is true or false

```
Do        {
//code
}while(condition);
```

3. for Loop in C: It also executes the code until condition is false. In this three parameters are given is

- Initialization
- Condition
- Increment/Decrement

```
for (initialization; condition; increment/decrement) {
 // Code statements to be executed
}
```

- **Break Statement:**The break statement is used to terminate the execution of a loop or switch statement. When encountered, it immediately exits the loop or switch and transfers control to the statement following the loop or switch.

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 1; i <= 10; i++) {
        if (i == 5) {
            break;  // Exit the loop when i equals 5      }
        printf("%d ", i);    }
    return 0; }
```

- “for” loop iterates from 1 to 10. However,
- when the value of “i” becomes 5, the “break” statement is encountered, and the loop is terminated prematurely. As a result, only the numbers 1, 2, 3, and 4 will be printed.

- **Continue Statement:** The continue statement is used to skip the remaining statements inside a loop and move to the next iteration of the loop. When encountered, it jumps to the loop’s condition evaluation and proceeds with the next iteration.

```
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i <= 10; i++) {
        if (i == 5) {
  continue;  // Skip the remaining statements in this iteration      }
        printf("%d ", i);    }
    return 0; }
```

- when the value of “i” is 5, the “continue” statement is encountered.
-This causes the program to skip the remaining statements within the current iteration of the loop and proceed to the next iteration.
-As a result, the number 5 is skipped, and the loop continues with the numbers 1, 2, 3, 4, 6, 7, 8, 9, and 10 being printed.

- **Goto Statement:** The goto statement is used to transfer control to a labeled statement within the same function. It allows the program execution to jump to a different section of code based on a specified label.

```
#include <stdio.h>
int main() {
        int i;
    for (i = 1; i <= 10; i++) {
        if (i == 5) {
         goto skip;  // Transfer control to the "skip" label      }
  printf("%d ", i);    }
    skip:    printf("\nSkipped number 5.");
 return 0; }
```

Output: 1 2 3 4

Skipped number 5.

-when the value of “i” is 5, the “goto” statement is encountered.
- It transfers the control of the program to the label named “skip”.
- As a result, the remaining statements within the current iteration of the loop are skipped, and the program directly proceeds to the “skip” label. The label is defined using the syntax label_name.

1. Which are not looping structures?

a. For loop b. while loop c. Do .. While loop d. if …else

2. How many times the following code prints the string “hello”

```
for(i=1;i<=50;i++)
Printf(“hello”);
```

a. 1 b. 50 c. Zero d. none of them

3. The first expression in a for… loop is

a. Step value of loop b. value of the counter variable

c. Condition statement d. none of the above

4. Which among the following is a unconditional control structure.

a. Goto b. for c. do-while d. if-else

For more practice: https://gtu-mcq.com/BE/Civil-Engineering/Semester-1/3110003/3815/MCQs?q=9aZHDjblmRk=

User defined function

```
return_type function_name( parameter list) {
        body of the function
}
```

- The ***return_type*** can be any valid C data type, and - the ***function_name*** can be any valid identifier

- The parameter list specifies the arguments that the function takes as input

- the body of the function contains the statements that are executed when the function is called.

For more Practice: https://gtu-mcq.com/BE/Civil-Engineering/Semester-1/3110003/3817/MCQs?q=9aZHDjblmRk=

Types of User-Defined Functions
1. Function with return value
2. Function without return value

1. Function have_____
a. Local b. Block. C. file. d. No
2. What is the default return value if it is not specified in function definition?
a. 0 b. -1 c. 1 d. none

Recursive Function:

- A function call itself is known as a recursive function.

1. A recursive function can be replaced with __ in C

a. For b. while c. do while d. all of these

2. A recursive function without if and else conditions will always lead to?

a. Finite loop b. infinite loop c. incorrect result d. correct result

3. **What is the output?**

```
int main() {
        int sum(int);
        int b;
        b = sum(4);
        printf("%d", b);  }
int sum(int x){
   int k=1;
   if(x<=1)
    return 1;
   k = x + sum(x-1);
   return k;}
```

a. 10 b. 11 c. 12 d. 15

4. What is the output?

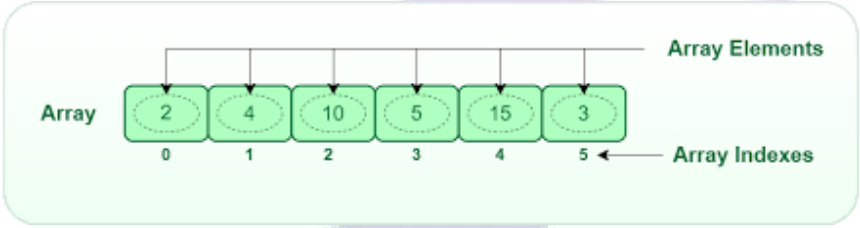
```
int mul(int);
int main()
{
    int b;
    b = mul(3);
    printf("%d", b);
}
int mul(int x)
{
    if(x<=1)
     return 1;
    return (x * mul(x-1));
}
```

a. 2 b. 3 c. 6 d. 1

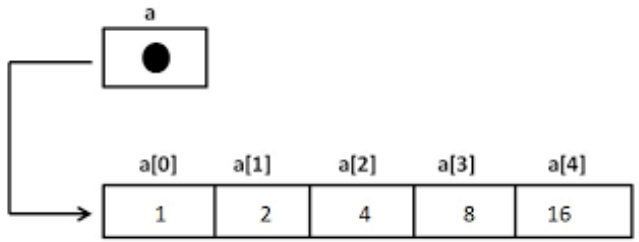
For more: https://gtu-mcq.com/BE/Civil-Engineering/Semester-1/3110003/3818/MCQs?q=9aZHDjblmRk=

Array:

- An array is a group of similar elements or data items of the same type collected at contiguous memory locations.



- Initialization of Array:



Declaration Syntax of Array:

- **VariableType VariableName[Sequence of Elements];**
- Example 1: For integral value
- int A[10];
- Here 10 means, this array A can have 10 integer elements.

5	12	45	67	1	2	3	4	7	9

- **Initialization of an Array:**

Syntax: datatype Array_Name[size] = { value1, value2, value3, ... valueN };

- **Types of Arrays:**
  - One-Dimensional Arrays
  - Multi-Dimensional Arrays

- Syntax for one dimensional : DataType ArrayName [size];
- For example: int a[10];
- Syntax: DataType ArrayName[row_size][column_size];
- For Example: int arr[5][5];
- Syntax: DataType ArrayName[size1][size2][size3];
- For Example: int a[5][5][5];

1. Assuming int is of 4bytes, what is the size of int arr[15];?

a. 30 b. 15 c. 4 d. 60

2. What is the output?

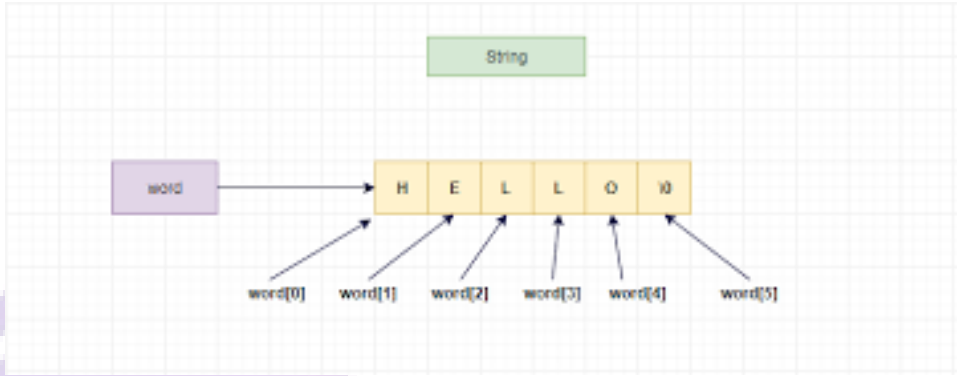
```
void main()
{
    int a[4]={5,6,7,8};
    printf("%d",a[1]);
}
```

a. 5 b. 6 c. 7 d. 8

For more practice: https://gtu-mcq.com/BE/Civil-Engineering/Semester-1/3110003/3816/MCQs?q=9aZHDjblmRk=

String:

- A string in C is like an array of characters, pursued by a NULL character.
- For example: char c[] = “c string”



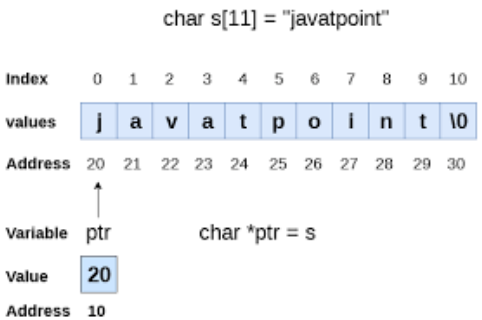
- **Syntax to Define the String in C**
- char string_variable_name [array_size];
- char c[] = “c string”
- **How to Declare a String?**

1. By Char Array

- char greeting[11] = {‘j’, ‘a’, ‘v’, ‘a’, ‘p’,’o’,’i’,’n’,’t’, ‘\0’};

2. By String Literal

- If you follow the rule of array initialisation, you can write the above statement as follows:
- char greeting[] = “javapoint”;



- **Function and Purpose of String:**
- strcpy(s1, s2); – It is used to copy string s2 into string s1.
- strcat(s1, s2); – It is used to concatenate string s2 onto the end of string s1.
- strlen(s1); – It is used to return the length of string.
- strcmp(s1, s2); – It is used to analyse and compare the strings s1 and s2. Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
- strchr(s1, ch); – It is used to discover the foremost event or occurrence of a specified character within the actual string.
- strstr(s1, s2); – It is used to return a pointer to a character at the first index

1. The format specifier to accept a string is

a. %d b. %c c. %f d. %s

2. If the two strings are identical, then strcmp() function returns

a. -1 b. 1 c. 0 d. true

3. What is the output?

```
void main()
{
    char str1[20]="Hello",str2[20]=" World";
    printf("%s\n", strcpy(str1, strcat(str1, str2)));
}
```

a. Hello b. Hello World c. WorldHellod. None of These

For more practice: https://gtu-mcq.com/BE/Civil-Engineering/Semester-1/3110003/3816/MCQs?q=9aZHDjblmRk=