**C Training**

**Data Types**

A screenshot of a computer

Description automatically generated

A screenshot of a white table with numbers and text

Description automatically generated

A white grid with black numbers

Description automatically generated

**Types of C Literals**

In C, Literals are the Constant values that are assigned to the constant variables.

There are 4 types of literals in C:

* **Integer Literal: decimals, octal-literals (045), hex-literals(0x23A), binary-literals(0b101)**
* **Float Literal**
* **Character Literal**
* **String Literal (**char stringVal[] = "GeeksforGeeks"**)**

# **Escape Sequence in C**

The escape sequence in C is the characters or the sequence of characters that can be used inside the string literal.

A screenshot of a phone

Description automatically generatedA screenshot of a computer

Description automatically generated

## Boolean in C

In C, the bool data type is not a built-in data type. It can be implemented in C using different methods as mentioned below:

1. Using header file “stdbool.h”
2. Using Enumeration type
3. Using define to declare boolean values

Boolean in C has the size of 1 byte as it needs only two values 0 and 1.

# **Integer Promotions in C**

Some data types like *char*, *short int*take a smaller number of bytes than *int*, these data types are automatically promoted to *int*or *unsigned int* when an operation is performed on them. This is called **integer promotion**. For example, no arithmetic calculation happens on smaller types like *char*, *short*and *enum*. They are first converted to *int*or *unsigned int*, and then arithmetic is done on them. If an *int*can represent all values of the original type, the value is converted to an *int*. Otherwise, it is converted to an *unsigned int.*

# **Character Arithmetic in C**

Character arithmetic is used to implement arithmetic operations like addition, subtraction, multiplication, and division on characters in C language. In character arithmetic character converts into an integer value to perform the task. For this ASCII value is used. It is used to perform actions on the strings.

# **Type Conversion in C**

## 1. Implicit Type Conversion

**-** Done by the compiler on its own, without any external trigger from the user.

**-** Generally, takes place when in an expression more than one data type is present. In such conditions type conversion (type promotion) takes place to avoid loss of data.

- bool -> char -> short int -> int -> unsigned int -> long -> unsigned -> long long -> float -> double -> long double

## 2. Explicit Type Conversion

This process is also called type casting and it is user-defined. Here the user can typecast the result to make it of a particular data type.

# **Basic Input and Output in C scanf(). printf()**

### **printf**("\n\nEnter the Sentence: "); **scanf**("%[^\n]\ns", str); // input with spaces

C language does not provide a format specifier for binary numbers.

## List of Format Specifiers in C

A screenshot of a document

Description automatically generatedA screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

# **Operators in C**

A diagram of numbers and symbols

Description automatically generated

## ****Bitwise Operators in C****

A screenshot of a computer

Description automatically generatedA white grid with black text

Description automatically generated

**Logical operators**

*The precedence of logical operators is: NOT, AND, OR.*

# **Increment and Decrement Operators in C**

### **Pre-Increment**

result = ++var1 is the same as var = var + 1; result = var

### **Post-Increment**

result = var1++ is the same as result = var; var = var + 1

## Syntax of Conditional/Ternary Operator in C

## The ternary operator in C is a conditional operator that works on three operands. It works similarly to the if-else statement and executes the code based on the specified condition. It is also called conditional Operator

*variable = Expression1* **?** *Expression2* **:** *Expression3****;***

*(condition)* **?** *(variable = Expression2)* **:** *(variable = Expression3)*;

# **sizeof operator in C**

sizeof() is a compile-time operator

**To find out the number of elements in an array:**

#include <stdio.h>

int main()

{

int arr[] = { 1, 2, 3, 4, 7, 98, 0, 12, 35, 99, 14 };

printf("Number of elements:%lu ",

sizeof(arr) / sizeof(arr[0]));

return 0;

}

**Data structures**

# **C Arrays**

An array in C is a fixed-size collection of similar data items stored in contiguous memory locations. The C arrays are static in nature, i.e., they are allocated memory at the compile time.

Array declaration: data\_type array\_name [size];

Array initialization: data\_type array\_name [size] = {value1, value2, ... valueN};

### **Array of Characters (Strings)**

In C, we store the words, i.e., a sequence of characters in the form of an array of characters terminated by a NULL character. These are called strings in C language.

**int** main()

{

    // creating array of character

**char** arr[6] = { 'G', 'e', 'e', 'k', 's', '\0' };

    // printing string

**int** i = 0;

**while** (arr[i]) {

**printf**("%c", arr[i++]);

    }

**return** 0;

}

## Properties of Arrays in C

### 1. Fixed Size

### 2. Homogeneous Collection

We can only store one type of element in an array. There is no restriction on the number of elements but the type of all of these elements must be the same.

### 3. Indexing in Array

The array index always starts with 0 in C language. It means that the index of the first element of the array will be 0 and the last element will be N – 1.

### 4. Dimensions of an Array

A dimension of an array is the number of indexes required to refer to an element in the array. It is the number of directions in which you can grow the array size.

### 5. Contiguous Storage

All the elements in the array are stored continuously one after another in the memory. It is one of the defining properties of the array in C which is also the reason why random access is possible in the array.

### 6. Random Access

The array in C provides random access to its element i.e we can get to a random element at any index of the array in constant time complexity just by using its index number.

### 7. No Index Out of Bounds Checking

There is no index out-of-bounds checking in C/C++, for example, the following program compiles fine but may produce unexpected output when run.

# **Pass Array to Functions in C**

*Arrays in C are always passed to the function as pointers pointing to the first element of the array.*

# **Bitmasking In C**

A bit is the smallest unit of data which can either store a 0 or 1 inside it

1. Setting a Bit

number **|** (1 **<<** bit\_position\_to\_set)

1. Clearing a Bit

number **& ~**(1 **<<** bit\_position\_to\_clear)

1. Flipping a bit

number ^ (1 << bit\_position\_to\_flip)

**Bitwise operators**

1. The **& (bitwise AND)** in C takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
2. The **| (bitwise OR)** in C takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
3. The **^ (bitwise XOR)** in C takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
4. The **<< (left shift)** in C takes two numbers, the left shifts the bits of the first operand, and the second operand decides the number of places to shift.
5. The **>> (right shift)** in C takes two numbers, right shifts the bits of the first operand, and the second operand decides the number of places to shift.
6. The **~ (bitwise NOT)** in C takes one number and inverts all bits of it.