Array:

What is array? how array is defined in c language?

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

**Declaring Arrays**

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

type arrayName [ arraySize ];

This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement −

double balance[10];

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

```c
#include <stdio.h>
 #include <conio.h>


int main()
{
    int a[1000],i,n,min,max;

    printf("Enter size of the array : ");
    scanf("%d",&n);

    printf("Enter elements in array : ");
    for(i=0; i<n; i++)
    {
        scanf("%d",&a[i]);
    }

    min=max=a[0];
    for(i=1; i<n; i++)
    {
         if(min>a[i])
          min=a[i];
           if(max<a[i])
            max=a[i];
    }
     printf("minimum of array is : %d",min);
          printf("\nmaximum of array is : %d",max);


    return 0;
}
```

```
Enter size of the array: 5
Enter elements in array: 1
2
3
4
5
minimum of an array is: 1
maximum of an array is: 5
```

**Why Do We Need Functions in C Programming?**

We need functions in C programming and even in other programming languages due to the numerous advantages they provide to the developer. Some of the key benefits of using functions are:

- Enables reusability and reduces redundancy
- Makes a code modular
- Provides abstraction functionality
- The program becomes easy to understand and manage
- Breaks an extensive program into smaller and simpler pieces

**Basic Syntax of Functions**

The basic syntax of functions in C programming is:

return_type function_name(arg1, arg2, … argn){

Body of the function //Statements to be processed

}

# Factorial:

```c
#include<stdio.h>
#include<math.h>
int main()
{
    printf("Enter a Number to Find Factorial: ");
    printf("\nFactorial of a Given Number is: %d ",fact());
    return 0;
}
int fact()
{
    int i,fact=1,n;
    scanf("%d",&n);
    for(i=1; i<=n; i++)
    {
        fact=fact*i;
    }
    return fact;
}
```

## Min and Max using functions in C Program Description:

```c
#include<stdio.h>

int min(int num1, int num2)
{
    return (num1 > num2) ? num2 : num1;
}

int main()
{
    int num1, num2, minimum;
    printf("Enter two numbers: ");
    scanf("%d%d", &num1, &num2);
    minimum =  min(num1, num2);
    printf("The minimum of %d and %d is : %d\n", num1, num2, minimum);
     return 0;
}
```

# Pointer:

The syntax of pointers is similar to the variable declaration in C, but we use the **( * ) dereferencing operator** in the pointer declaration.

`datatype * ptr;`

where

- **ptr** is the name of the pointer.
- **datatype** is the type of data it is pointing to.

The above syntax is used to define a pointer to a variable. We can also define pointers to functions, structures, etc.

Why we use Pointer?

Pointers are used extensively in both C and C++ for three main purposes: **to allocate new objects on the heap, to pass functions to other functions**. **to iterate over elements in arrays or other data structures**. Program:

```c
#include <stdio.h>
int main()
{
  int var = 5;
  printf("var: %d\n", var);

  // Notice the use of & before var
  printf("address of var: %p", &var);
  return 0;
}
```

```
var: 5
address of var: 2686778
```

Character and String:

How a character and string variable is declared in C programming Language?

**In C programming, a character variable can hold a single character enclosed within single quotes. To declare a variable of this type, we use the keyword char, which is pronounced as kar. With a char variable, we can store any character, including letters, numbers, and symbols.**

Declaring a string in C is as simple as declaring a one-dimensional array. Below is the basic syntax for declaring a string.

```
char string_name[size];
```

In the above syntax **str_name** is any name given to the string variable and size is used to define the length of the string, i.e the number of characters strings will store.

There is an extra terminating character which is the **Null character ('\0') used to indicate the termination of a string that differs strings from normal character arrays**.

**Program to compare two strings and print alphabetical order?**

```c
#include <stdio.h>
#include <string.h>
 int main()
{
    char s1[1000],s2[1000];
    int i,c=0;
     printf("Enter  string1: ");
    gets(s1);
    printf("Enter  string2: ");
    gets(s2);
    if(strlen(s1)==strlen(s2))
    {
        for(i=0;s2[i]!='\0';i++)
        {
            if(s1[i]==s2[i])
             c++;
        }
        if(c==i)
          printf("strings are equal");
        else
          printf("strings are not equal");
    }
    else
     printf("strings are not equal");
     return 0;
}
```

```
/tmp/CiMv46VWNR.o
Enter  string1: hello
Enter  string2: hello
strings are equal
```

Program to read a string and display the length of the string?

```c
#include <stdio.h>
int main() {
    char s[] = "Programming is fun";
    int i;

    for (i = 0; s[i] != '\0'; ++i);

    printf("Length of the string: %d", i);
    return 0;
}
```

```
Length of the string: 18
```

Structure:

Why structure variable is needed?

Structure in C programming is **very helpful in cases where we need to store similar data of multiple entities**. Let us understand the need for structures with a real-life example. Suppose you need to manage the record of books in a library. Now a book can have properties like book_name, author_name, and genre.

define a struct type book_bank variable to store book information?

File:Program to read an array from keyboard and store the values into

```c
#include <stdio.h>
int main()
{
    int a[1000],i,n;
     printf("Enter size of array: ");
    scanf("%d",&n);
     printf("Enter %d elements in the array : ", n);
    for(i=0;i<n;i++)
    {
        scanf("%d", &a[i]);
    }
    printf("\nElements in array are: ");
    for(i=0;i<n;i++)
    {
        printf("%d  ", a[i]);
    }
    return 0;
}
```