Arrays

1. What would happen if you try to put so many values into an array when you initialize it that the size of the array is exceeded?
2. Can the size of an array be changed after it is declared?
3. What is the difference between a single-dimensional and multi-dimensional array?
4. What are the default values of array elements if you don't initialize them?
5. What is the null character \0 in a C string?
6. Explain what happens if you try to access an element outside the array's range.

INPUT/OUTPUT QUESTIONS

1.
```
#include<stdio.h>

int main()

{

   int a[5] = {5, 1, 15, 20, 25};

   int i, j, m;

   i = ++a[1];

   j = a[1]++;

   m = a[i++];

   printf("%d, %d, %d", i, j, m);

   return 0;

}
```

Output

```
3, 2, 15
```

2.

```
#include <stdio.h>

int main() {

   int x = 10, y = 20;

   printf("%d\n", x > y ? x : y);

   return 0;

}
```

Answer: The output will be 20.

3

```
#include <stdio.h>

int main() {

   char str[] = "Hello";

   printf("%s\n", str + 1);

   return 0;

}
```

Answer: The output will be **ello**

4

```
#include <stdio.h>

int main() {

   int arr[] = {10, 20, 30, 40};

   printf("%d\n", arr[3] - arr[1]);

   return 0;

}
```

Answer:
Output: 20

Explanation:
arr[3] is 40 and arr[1] is 20. The difference is 40 - 20 = 20.

5.

```
void operationArray(int arr[], int size) {

   int start = 0;

   int end = size - 1;

   while (start < end) {

      int temp = arr[start];

      arr[start] = arr[end];

      arr[end] = temp;

      start++;

      end--;

   }

}


int main() {

   int arr[5] = {1, 2, 3, 4, 5};

   operationArray(arr, 5);

   for (int i = 0; i < 5; i++) {

      printf("%d ", arr[i]);

   }

   return 0;

}
```

Output: 5 4 3 2 1

6. int main() {

```
  int a = 5;
  int b = a--;
  printf("%d %d\n", a, b);
  return 0;
}
```

Ans : 4 5

Q7.

```
int main() {
  int arr[5] = {1, 2, 3, 4, 5};
  int i;
  for (i = 4; i > 0; i--) {
    printf("%d ", arr[i]);
  }
  printf("\n");
  return 0;
}
```

Ans: 5 4 3 2

Q8.

```
int main() {
    int arr[5] = {10, -20, 50, 80, 131};
    int max = arr[0];
    int i;
    for (i = 1; i < 5; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    printf("%d\n", max);
    return 0;
}
```

Ans: 131

Q9.

```
int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    int sum = 0;
    int i;
    for (i = 0; i < 5; i++) {
        sum += arr[i];
    }
    printf("%d\n", sum);
    return 0;
}
```

Ans: 15

==> Fill in the blanks

1.

```
int main() {
   int arr[3] = {1, 2, sizeof(int)};
   printf("%d %d %d\n", ______________);
   return 0;
}
```

Output: 1 2 4

Ans: arr[0], arr[1], arr[2]

2.
```
int main() {
   int arr[5] = {1, 2, 3, 4, 5};
   int min = arr[0];
   int i;
   for (i = 1; i < 5; i++) {
      if (___________) {
         min = arr[i];
      }
   }
   printf("%d\n", min);
   return 0;
}
```

Output: 1

Ans: arr[i] < min

Q3.

```
int sumOfSquares(int arr[], int n) {
   int sum = 0;
   int i;
   for (i = 0; i < n; i++) {

       ________________________
   }
   return sum;
}

int main() {
   int arr[3] = {1, 2, 3};
   int result = sumOfSquares(arr, 3);
   printf("%d\n", result);
   return 0;
}
```

Output: 14

Ans: sum += arr[i] * arr[i];

Functions

Q1.

```
int multiply(int a, int b) {

    ____________________
}

int main() {
    int result = multiply(2, 3);
    printf("%d\n", result);
    return 0;
}
```

Output: 6

Ans: return a * b;

Q2.

```
int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

int main() {
    int result = max(-2, -3);
    printf("%d\n", result);
    return 0;
}
```

Ans: -2

Q3.

```
int func1(int a, int b) {
    return a * b;
}

int main() {
    int result = func1(4, 8);
    printf("%d\n", result);
    return 0;
}
```

Ans: 32

Q4.

```
void funcPrint(int arr[], int n) {
    int i;
    for (i = n-1; i >= 0; i--) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[3] = {1, 2, 3};
    funcPrint(arr, 3);
    return 0;
}
```

Ans: 3 2 1

1. What is a function? Why we use functions in C language? Give an example.

Ans: Function in C:

A function is a block of code that performs a specific task. It has a name and it is reusable .It can be executed from as many different parts in a program as required, it can also return a value to calling program.

All executable code resides within a function. It takes input, does something with it, then give the answer. A C program consists of one or more functions.

A computer program cannot handle all the tasks by itself. It requests other program like entities called functions in C. We pass information to the function called arguments which specified when the function is called. A function either can return a value or returns nothing. Function is a subprogram that helps reduce coding.

Simple Example of Function in C

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

```
#include <conio.h>

int adition (int, int); //Function Declaration int

addition (int a, int b) //Function Definition

{

int r;

r=a + b;

return (r);

}

int main(){

int z;

z= addion(10,3); //Function Call

printf ("The Result is %d", z);

return 0;

}

Output: The Result is 13
```

Q.1. **Why use function:**?

Basically there are two reasons because of which we use functions

1. Writing functions avoids rewriting the same code over and over. For example - if you have a section of code in a program which calculates the area of triangle. Again you want to calculate the area of different triangle then you would not want to write the same code again and again for triangle then you would prefer to jump a "section of code" which calculate the area of the triangle and then jump back to the place where you left off. That section of code is called „function'.

2. Using function it becomes easier to write a program and keep track of what they are doing. If the operation of a program can be divided into separate activities, and each activity placed in a different function, then each could be written and checked more or less independently. Separating the code into modular functions also makes the program easier to design and understand

Q. 2. Distinguish between Library functions and User defined functions in C and Explain with examples.

Ans: Types of Function in C: (i). Library Functions in C

C provides library functions for performing some operations. These functions are present in the c library and they are predefined.

For example sqrt() is a mathematical library function which is used for finding the square root of any number .The function scanf and printf() are input and output library function similarly we have strcmp() and strlen() for string manipulations. To use a library function we have to include some header file using the preprocessor directive #include.

For example to use input and output function like printf() and scanf() we have to include stdio.h, for math library function we have to include math.h for string library string.h should be included.

(ii). User Defined Functions in C

A user can create their own functions for performing any specific task of program are called user defined functions. To create and use these function we have to know these 3 elements.

1. Function Declaration

2. Function Definition

3. Function Call

1. Function declaration

The program or a function that calls a function is referred to as the calling program or calling function. The calling program should declare any function that is to be used later in the program this is known as the function declaration or function prototype.

2. Function Definition

The function definition consists of the whole description and code of a function. It tells that what the function is doing and what are the input outputs for that. A function is called by simply writing the name of the function followed by the argument list inside the parenthesis.

Function definitions have two parts:

Function Header

The first line of code is called Function Header.

int sum(int x, int y)

It has three parts

(i). The name of the function i.e. sum

(ii). The parameters of the function enclosed in parenthesis

(iii). Return value type i.e. int

Function Body

Whatever is written with in { } is the body of the function.

3. Function Call

In order to use the function we need to invoke it at a required place in the program. This is known as the function call.

Q.3. Explain the various categories of user defined functions in C with examples? (Discussed in the lecture)

Q.4. Explain the Parameter Passing Mechanisms in C-Language with examples. (Discussed in the lecture)

Q.5. What is actual parameters and formal parameters. Explain with Example. (Discussed in the lecture)