Array in C and Fortran

For One Dimensional Array		
	C	FORTRAN
DECLARATION	int a[50]	INTEGER A(50) DIMENSION A(50)

INPUT	for(i=0;i<n;i++)
	{
	scanf("%d", &a[i]);
	}
		DO 100 I=1, N
		READ(*,*) A(I)
		100 CONTINUE

INPUT USING IMPLIED LOOP		READ(*,*)(A(I), I = 1, N)
PROCESSING	Single ***for*** loop in general but may vary based on question , use a[i]	SINGLE ***DO*** LOOP IN GENERAL BUT MAY VARY BASED ON QUESTION , USE A(I)

OUTPUT	for(i=0;i<n;i++)
	{
	printf("%d", a[i]);
	}
		DO 100 I=1, N
		WRITE(*,*) A(I)
		100 CONTINUE

OUTPUT USING IMPLIED LOOP		WRITE(*,*)(A(I), I = 1, N)
For Two Dimensional Array		

INPUT	for(i=0;i<m;i++)
	{
	for(j=0;j<n;j++)
	{
	scanf("%d", &a[i][j]);
	}
	}
		DO 100 I=1,M
		DO 200 J=1,N
		READ(*,*)A(I,J)
		200 CONTINUE
		100 CONTINUE

INPUT USING IMPLIED LOOP		DO 300 I=1, M READ(*,*)(A(I, J), J=1, N) 300 CONTINUE
PROCESSING	Nested loop is required in general, but can vary based on the problem, use a[i][j]	Nested loop is required in general, but can vary based on the problem, use A(I, J)

OUTPUT	for(i=0;i<m;i++)
	{
	for(j=0;j<n;j++)
	{
	printf("%d ", a[i][j]);
	}
	}
		DO 100 I=1,M
		DO 200 J=1,N
		WRITE(*,*)A(I,J)
		200 CONTINUE
		100 CONTINUE

OUTPUT USING IMPLIED LOOP		DO 400 I=1,M WRITE(*,*) (C(I, J), J=1, N) 400 CONTINUE

For One Dimensional Array		
	ARRAY	POINTER
DECLARATION	int a[50];	int *a;

INPUT	for(i=0;i<n;i++)
	{
	scanf("%d", &a[i]);
	}
		for(i=0;i<n;i++)
		{
		scanf("%d", a+i);
		}

PROCESSING / FUNCTION DEFINITION	Single loop, a[i]	Single loop, *(a+i)

OUTPUT	for(i=0;i<n;i++)
	{
	printf("%d", a[i]);
	}
		for(i=0;i<n;i++)
		{
		printf("%d", *(a+i));
		}

FUNCTION DECLARATION	void sort(int [], int);	void sort(int *, int);
FUNCTION CALL	sort(a, n)	sort(a, n)
For Two Dimensional Array		
DECLARATION	int a[5][5];	int **a;

INPUT	for(i=0;i<m;i++)
	{
	for(j=0;j<n;j++)
	{
	scanf("%d", &a[i][j]);
	}
	}
		for(i=0;i<m;i++)
		{
		for(j=0;j<n;j++)
		{
		scanf("%d", (*(a+i)+j));
		}
		}

PROCESSING / FUNCTION DEFINITION	Nested loop is required in general, but can vary based on the problem, use a[i][j]	USE *(*(a + i) + j)

OUTPUT	for(i=0;i<m;i++)
	{
	for(j=0;j<n;j++)
	{
	printf("%d ", a[i][j]);
	}
	}
		for(i=0;i<m;i++)
		{
		for(j=0;j<n;j++)
		{
		printf("%d ", *(*(a + i) + j));
		}
		}

FUNCTION DECLARATION	void add(int [][5], int[][5], int[][5], int, int);	void add(int**, int**, int**, int, int);
FUNCTION CALL	add(a, b, c, m, n);	add(a, b, c, m, n);

```
C     Program to add the elements of two matrices

      integer a(5,5), b(5,5),c(5,5), i, j, m, n, p, q
      write(*,*)'Enter order of first matrix '
      read(*,*)m,n
      write(*,*)'Enter order of second matrix '
      read(*,*)p,q

      if(m.eq.p .and. n.eq.q) then
      write(*,*)'Enter elements of first matrix '

      do 100 i=1,m
        do 200 j=1,n
          read(*,*)a(i,j)
200     continue
100   continue

      write(*,*)'Enter elements of second matrix
      do 300 i=1, p
         read(*,*)(b(i,j), j=1, q)
300   continue

      do 500 i=1,m
        do 600 j=1,n
          c(i,j)=a(i,j) + b(i,j)
600     continue
500   continue

      write(*,*)'Final Elements are '

      do 400 i=1,m
        write(*,*)(c(i,j), j=1, n)
400   continue

      else
        write(*,*)'Invalid order'
      end if

      pause
      stop
      end
```

```
C     Program to insert 3 elements at the
C     middle of an array with n elements

         integer a(50), b(3), c(60) ,i , n, pos, ee
         write(*,*)'How many elements '
         read(*,*)n
         write(*,*)'Enter elements '

         do 100 i=1,n
                read(*,*)a(i)
100      continue

      ee = 3
      write(*,*)'Enter elements to be inserted '
      read(*,*)(b(i),i=1, ee)

      pos = n/2
         do 200 i=1, n+ee
                if(i .le. pos) then
                       c(i) = a(i)
                else if(i .le. (pos + ee)) then
                       c(i) = b(i-pos)
                else
                       c(i) = a(i-ee)
                end if
200      continue

         write(*,*)'Final Updated array is '

         write(*,*)(a(i),i=1,n)
         write(*,*)(b(i),i=1,ee)


         do 400 i=1,n+ee
                write(*,*)c(i)
400      continue

         pause
         stop
         end
```

1. **Conversion samples from formatted to unformatted I/O functions**

Type and Declaration	Formatted	Unformatted	Remarks
String char name[50];	printf("Enter any text : ");	puts("Enter any text : ");	puts() adds newline after displaying its content

	printf("Name : %s", name);	puts("Name : ");
		puts(name);
			printf() displays its content in a single line unless \n is used

	scanf("%s", name);	gets(name);	scanf() with %s as format specifier does not read characters after space, so gets() can be used to read a string or text with spaces

Character
char c;
	printf("Character : %c", c);	puts("Character : ");
		putchar(c);
			puts() is used to display the information and putchar() is used to display value of character variable.

	scanf("%c", &c);	c =getchar();	

2. **Convert the following into unformatted I/O functions.** (*Write multiple statements in one line*)

printf("Enter your section : ");	
printf("Section is %c", section);	
printf("Name is %s\nSection is %c", n, c);	
printf("%s is my address", add);	
scanf("%c", &ad);	
scanf("%s", rl);	

3. **Rewrite the following code using Unformatted I/O functions.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char name[50];
        char grade;
        printf("Enter your name : ");
        scanf("%s", name);
        printf("Enter your grade : ");
        scanf("%c", &grade);
        printf("Name : %s", name);
        printf("\nGrade : %c", grade);
        getch();
}
```