Well Known as Chapter 3. by C.S.E. Students

DELTA Page No. Date / @shubhamdreamwork

22/12/18.

ARRAY (1-D, 2-D)

=> An Array is a collection of items of same data type stored at contiguous memory locations.

- Array in C are Container types that Contain a finite number of homogenous elements

They are of static type and hence the size of array cannot be altered at runtime

TYPES OF ARRAY

1.) (1-D Array)

Syntax: arrayName[index];

=> [nth element address = base address of array + (n * size of elements in bytes)]

Example.

```
#include<stdio.h>
int main()
{ int arr[5] = {1,2,3,4,5};
  for(int=i; i<5; i++)
    { Printf("Value of %d location is : %d \n");
      } return 0;
}
```

2.) (2-D Array) or (Multi Dimensional Array)

Two-dimensional Arrays are indexed by two subscripts one for the row and one for the Column.

Data Type ArrayName[MAX_ROWS][MAX_COL];

:. String:- A String is a group of Characters of any length The String Enclosed within double quotation mark is known as literal.

/* Passing Parametres to function in C */
(Parameters are input given by the user to particular function)

⇒ There are two types of Parameter Passing Techniques
- Call by value
- Call by Reference

Before going to the Concept of Parameter Passing Let's Understand Actual Parameters and Formal Parameters with Example

```
void disp(int x);  → Formal Parameter
main()
{ int a = 100;
  disp(a);  → Actual Parameter.
```

In Real,
Actual Parametres are function Call
Formal Parametres are Function Definition.

Let's Understand Passing technique in short by Examples of Swapping of Elements

- Call by Value

In Call by Value Actual Parametres are Copied to Formal Parametres making a new different memory location for new item.

Example.

```
void swap(int, int) /* function difinition*/
main()
{ int a=20, b=30;
  swap(a,b); /* function Call */
  Printf("a=%d\t b=%d", a, b);
}
```

```
void swap (int x, int y)
{ int temp;
  temp = x;
  x = y;
  y = temp;
  Printf("x = %d \t y = %d", x, y);
}
```

- Call by Reference

Actual Parameter and Formal Parameter shares same memory location.

```
void swap (int *, int *)
main()
{ int a=20; b=30;
  swap (&a, &b)
  Printf("a = %d \t b = %d", a, b);
}

void swap (int *a, int *b)
{ int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
```

Recursion as a different way of solving Problems

Function Calling itself is Called Recursive function.

And, Calling a function from definition of same function is known as Recursion.

Let's Understand, It with the taking Example of a Program to Compute factorial.

```
main()
{ int n, res;
  Printf("Enter n value");
  Scanf("%d", &n);
  res = fact(n);
  printf("result : %d", res);
}

fact(int n)
 { int res;
   if(n==0)
    res=1;
   else
    res = n * fact(n-1);
    return res;
 }
```

Passing Array through Pointers

=> Initialising `int *P = &arr[0];`] Any one can be used
`int *P = arr`
`P++;`

=> For pointing base address of an array { arr[0] or arr }

```
main()        => Please Include necessary Header files
{ int arr[5], i, total = 0; int *P;
  Printf("Enter five elements");
  for(i=0; i<5; i++)
   { Scanf("%d", &arr[i]); }
    *P = arr;
   Printf("Elements are :");
   for(i=0; i<5; i++)
    { Printf("%d", *P);
      total = total + *P;
      P++;
    } Printf("Total = %d", Total);
      getch();
 }
```

PASSING ARRAYS TO FUNCTIONS

```
void func(int a[], int size)
{
}
int main()
{
    int n=5;
    int arr[5] = {1,2,3,4,5};
    func(arr, n);
    return 0;
}
```

Here, Pointer a tells base address of array arr.

Size or Length of array is Passed.

Example:

```
#include<stdio.h>
void fun(int arr[], int n)
{ int i;
  for(i=0; i<n; i++)
    Printf("%d", arr[i]);
} // Driver Program

int main()
{
  int arr[] = {1,2,3,4,5,6,7,8};
  int n = size of(arr) / size of(arr[0]);
  fun(arr, n);
  return 0;
}
```

When we Pass an array to a function say fun(), It is always treated as a pointer by fun()

There are 3 ways to Passing, It's my favourable Part.

INPUT/OUTPUT FUNCTION

Function	Type	Description
getchar()	Stream	Read a Char from Stream
putchar()	Stream	Write a Char to Stream
getc()	Stream	Read a Char from Stream
putc()	Stream	Write a Char from Stream
gets()	Stream	Read a String from Stream
puts()	Stream	Write a String to Stream
cgets()	Console	
getche()	Console	
putch()	Console	
cputs()	Console	

LIBRARY FUNCTIONS

```
strcpy (S1, S2);
strlen (S1);
strcmp (S1, S2);
strchr (S1, ch);
strstr (S1, S2);
```

Library is a set of Header files

stdio.h	conio.h	string.h
→ Printf();	Clrscr();	Strlen();
→ Scanf();	Getche();	Strrev();

Graphics.h	math.h	Das.h
→ setcolor();	rand();	getdata();
→ setbkcolor();	Pow();	gettime();

A Library is a set of Header files and header file is set of Predefined functions.

बुधवार, ३० मई • ज्येष्ठ (अधिक) बदी १ • शाका: ६ ज्येष्ठ • विक्रम १७ ज्येष्ठ

Previous Date	Case No.	Name of the Court	Party Name V/S	Position Stage	Next Date

-> C Program for Fibonacci series using a loop ~~and Recursion~~

=> The first few numbers of the series are 0, 1, 1, 2, 3, 5, 8. Except first two terms other terms is the sum of previous two term.

```
# include <stdio.h>
int main()
{ int n, first=0, second = 1, next, c;

  Printf ("Enter the number of term\n");
  Scanf ("%d :", &n);

  Printf ("First %d terms of fibonacci series are:
          \n", n);

  for (c=0; c<n; c++)
  { if (c<=1)
      next = c;
    else
    {
      next = first + second.
      first = second;
      second = next;
    }
  Printf ("%d\n", next);
  }
    return 0;
}
```

THE FUNCTION OS WISDOM IS DISCRIMINATING BETWEEN GOOD AND EVIL.

Previous Date	Case No.	Name of the Court	Party Name V/S	Position Stage	Next Date

Ackermann function is a classic example of a recursive function, notable especially because it is not a primitive recursive function. It grows very quickly in value, as does the size of its call tree.

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m, n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

```
# Include <stdio.h>
int ackermann(int m, int n)
{
    if (!m) return n+1;
    if (!n) return ackermann(m-1, 1);
    return ackermann(m-1, ackermann(m, n-1));
}
int main()
{
    int m, n;
    for (m=0; m<=4; m++)
        for (n=0; n<6-m; n++)
            printf("A(%d, %d) = %d\n", m, n,
                   ackermann(m, n));
    return 0;
}
```

Case No.	Name of the Court	Party Name V/S	Position Stage

Output:

A(0,0) = 1.
A(0,1) = 2
A(0,2) = 3
A(0,3) = 4
A(0,4) = 5
A(0,5) = 6
A(1,0) = 2
A(1,1) = 3
A(1,2) = 4
A(1,3) = 5
A(1,4) = 6
A(2,0) = 3
A(2,1) = 5
A(2,2) = 7
A(2,3) = 9
A(3,0) = 5
A(3,1) = 13
A(3,2) = 29
A(4,0) = 13
A(4,1) = 65533