Arrays

(*) Do not attempt the following operations on pointers... they would never work out!-

(a). Addition of two pointers
(b). multiplication of a pointer with a constant.
(c). Division of a pointer with a constant.

(*). Accessing array element by pointers is much (or) always faster than accessing them by subscripts.

(*) base address can also be passed by just passing the name of the array.

(*) Two-D Array called matrix.

(*) While initialising a 2-D array it is necessary to mention the second (column) dimension, whereas the first dimension row is optional.

⊛ In memory whether it is one-D (or) 2-D array the array elements are stored in one continous chain.

⊛ Each row of a 2-D array can be thought of as a 1-D array

⊛ s[2][1]
*(s[2]+1)
((s+2)+1)

⊛ A more general formula for accessing each array element wolld be:-

$\boxed{\text{*(base address + row no. * no. of columns + column no.)}}$

Functions & Pointers

⊛ Break a program into small units and write functions for each of these subdivisions.

⊛ A function can return only one value at a time.

⊛ There are two possibilities for calling convention:-

ⓐ. Argument might be from left to right

ⓑ Argument might be from right to left.

C-language follow this

⇒ [From Right to left]

⊛ Any function by default returns an int value

⊛ **k = k is pointer to an integer pointer.

Structures

Functions:-

* A function cannot return two values at a time.
* If no value is returned in function so, there is nor need to collect it in variable a.
* variable should be declared as float and char respectively in the function printit ( ).
* In a function definition, semicolon should be present.
* one function cannot be defined within another function.
* No semicolon present after function in function definition.
* A function may contain more than one return ~~funct~~ statements.
* Same names cannot be used for different functions without any conflict.

* ~~## 06~~

```
/* obtain prime factors of a number */
#include <stdio.h>
void prime (int);
int main() {
    int num;
    printf ("Enter number:");
    scanf ("%d", &num);
    prime (num);          // function call
return 0;
}
void prime (int num) {
    int i=2;
    printf ("Prime factors of %d are ", num);
    while (num! =1) {
        if (num % i==0)
            printf ("%d ", i);
        else
        {
            i++;
            continue;
        }
        num = ~~num~~num /i;
    }
}
```

Pointers.

* The declaration of variables should be inside the brace
* Should be used semicolon in function declaration
* Standard deviation:-

$$\boxed{\sigma = \sqrt{\frac{\Sigma (x_i - \mu)^2}{N}}}$$

* function to evaluate a series:-

$\sin(x) = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + —$

```
/* evaluation of a series */
#include<stdio.h>
# include<math.h>
float numerator (float ,int);
float denominator (int);

int main()<
  float n, x, a, b, sum=0;
   int i,j;

  printf("\n Enter the number x: ");
  scanf(" %f", &x);
  for (i=1, j=1 ; i<=10; j++, j+=2) <
    /* upto 10 terms */
     a= numerator (x,j);
     b= denominator (j);
     n= a/b;
     (i % 2 ==0) ? sum = sum -n : (sum =sum +n);
  >
  printf("sum = %f\n", sum);
  return 0;
>

/* Calculate power */
float numerator (float y, int j)<
     float k=1;
     int m;
    for (m =1; m <=j ; m++)
          k* =y;

    return (k);
}

/* Calculate factorial */
float denominator (int j) {
      int m;
      float h=1;

       for (m=1 ; m <=j ; m++) {
             h=h* m;

             ~~ret~~
        return h;
     }
```

Array

* While declaring the 2D Array mentioning the column dimension necessary.

* literals → represents the fixed value.
* The header file provides the printf() function. It is used to print the string in quotation marks.
* The ~~ss~~ scanf() variable takes the input from the user and stores in a variable.

* %lf, %d, %c, %f ⇒ format specifier.

* int x=5;

  float a=x;

  data type of x remains the same. Only the value of x is converted to float when we assign x to a.
* True is represented by 1 in C.
* Logical AND - &&.
* The break statement terminates the loop in which it is used
* The name of the function is given right after the return type
* The function has int as a return type. So, we must return an int value to the function.
* The function prototype declares the function by specifying the function return ~~t~~ype, function name and function parameters:
* The standard library function have some predefined meaning.

* The num variable is a static variable whose value persists until the end of the program. Hence, in the second function call, the value of num will be 9 instead of 7.

* In recursive function, a function calls itself inside the body of the function.

* Default element in ~~arry~~ array is 0.

* we cannot have arrays as a return type for a function in C.

* int marks [5];
  marks gives the address of the first element of the array. Hence, &marks[0] is the same as marks.

* The stdlib.h contains standard utility functions that can be used for dynamic memory allocation.
  ex. - malloc(), calloc(), realloc() & free().

Structure - - - -

* Struct or Structure can be stored in array form.

* Age cannot be access directly.

* Give proper name to given structure.

* while printing signature & status, v[i].signature & v[i].status should be used.

* In function f() dot operator should be used to access structure elements, while g() → operator should be used to access structure elements.

* All structure elements are stored in contigous memory location.

* while declaring a two-D array mentioning the column dimension is necessary.

* An array cannot be declared to be of the type ~~of~~ in char.

Logical error, Array bounds are exceeded

- Dimensions of array should be constant and all declaration should be at the begining.

- Array is a collection of the same data type placed next to each other in memory.

- A preprocessor directive is a message from programmer to the preprocessor.

- function cannot return 2 values at a time.

- One function cannot be defined within another function.