Tutorial 1

Pointer tutorial

- 2d array as function argument.
- char pointer accessing int value

This is a function displaying elements of a 2D array.
Now, you modify this to display row-wise sum (or column-wise sum).

```
#include <stdio.h>
void print_2d_array(int *a, int rows, int cols);
void print_2d_array(int *a, int rows, int cols)
{
        for(int i = 0; i < rows; ++i) {
                for(int j = 0; j < cols; ++j) {
                printf("%d ", a[i * cols + j]);
                }
                printf("\n");
         }
}
int main(void)
{
int arr[][3] = { {1, 2, 3}, {4, 5, 6}  };
print_2d_array(arr[0], 2, 3); /*first argument type is what?? */
}
```

char pointer accessing int value

```
#include <stdio.h>
int main()
{
    int i = 23;
    char *p;
    p = (char *) &i;
    int s = sizeof(i)/sizeof(char);
    for (int k=0; k<s; k++) {
        printf("%d, ", *p); p++; }
    return 0;
}
```

my_rand( ) with void return;

- void my_rand(int *p)

```
{
        *p = rand( );
}
```

```
#include <stdio.h>
#include <stdlib.h>
void my_rand(int *p);

int main(){
	int i;
	my_rand(&i);
	for (int j=0; j<10; j++) {
		my_rand(&i);
		printf("%d, ", i);
	}
return 0;
}

void my_rand(int *p)
{
*p = rand();
}
```

void

- Funtion’s return type being void says that the function returns nothing.
- A pointer of type void

  void *p;
- This is a general purpose pointer.
- Now, p can be assigned with address of any type variable.
- But, to access the value through p using the dereferencing operator like *p will not work.
- You need to typecast it.

```
#include < stdio.h >
void main()
{
    int p=20;
    float q=15.75;
    void *ptr;     // Declaring a void pointer
    ptr=&p;        // Assigning address of integer to void pointer.
    printf("The value of integer variable is= %d",*( (int*) ptr) );
    // (int*)ptr - is used for type casting. Whereas *((int*)ptr)
    // dereferences the type casted void pointer variable.
    ptr=&q;     // Assigning address of float to void pointer.
    printf("The value of float variable is= %f",*( (float*) ptr) );
}
```