Array of Pointers

int *ptr[arraysize] = array of pointers, pointing to int

Pointer to function

```
int (*FuncPTR)(int a,int b);
//called funcptr is a pointer
to function
// actually is used as a
parameter of another func and
can pass any func to the desired
func as a parameter with this
method
int func1(int);
int func2(int);
int func3(int (*FuncPTR)(int));
now we can pass func1 or func2
to func3 ;
```

Array of Pointers to functions

```
void (*f[3])(int)=
{function1,function2,function3};
// f is an array of pointers ,
pointing to functions of type
void which all of them take one
parameter of type int .
```

Pointer vs array

```
int b[10]; int
*bptr;
bptr=b; OR
bptr=&b[0];
*(bptr+3) // shows
the value of b[3]
 bptr+3 // points
to &b[3]
//an array can be
used like a pointer
too -> *(b+3)-
=value of b[3]
//pointer to an
array can be used
like an array ->
bptr[3] = value of
b[3]
```

NULL pointer

```
int *ptr = NULL;
//The value of ptr
is 0
if(ptr) // succeeds
if p is not null
if(!ptr) //
succeeds if p is
null
```

passing pointers to functions

*WHEN PASSING
ARGUMENTS*

```
   unsigned long
sec;
   getSeconds(
&sec );
```

IN FUNCTION HEADER

```
void getSeconds(u-
nsigned long *par)
```

Pointer to pointer

```
A pointer to a pointer is a
form of multiple indirection or
a chain of pointers
    int var;
    int *ptr;
    int **pptr;
    var = 3000;
    ptr = &var; // take the
address of var
    pptr = &ptr;// take the
address of ptr using address of
operator &
```

Return pointer from functions

```
int * getRandom( ) {
    static int r[10];
    return r;
}
// main function to call above
defined function.
int main () {
    int *p;
    p = getRandom();
    }
```

Array of strings

```
char *color=
{"red","blue","yellow",'green'};
//color is an array of pointers
, pointing to the first
character of each string
```