

The background is a deep blue gradient with a subtle pattern of small white dots. Overlaid on this are several faint, light blue circular elements. These include concentric circles, some with arrows indicating a clockwise direction, and a large circular scale with numerical markings from 140 to 260 in increments of 10. The overall aesthetic is technical and futuristic.

DYNAMIC MEMORY ALLOCATION

PROGRAMMING LANGUAGES

BY

Z. CIHAN TAYSI

OUTLINE

- Memory allocation functions
- Array allocation
- Matrix allocation
- Examples

MEMORY ALLOCATION FUNCTIONS

- ***malloc()***
 - Allocates a specified number of bytes in memory. Returns a pointer to the beginning of the allocated block.
- ***calloc()***
 - Similar to *malloc()*, but initializes the allocated bytes to zero. This function allows you to allocate memory for more than one object at a time.
- ***realloc()***
 - Changes the size of a previously allocated block.
- ***free()***
 - Frees up memory that was previously allocated with *malloc()*, *calloc()*, or *realloc()*.

MEMORY ALLOCATION FUNCTIONS

- `void *malloc(size_t size);`
- `void *calloc(size_t nmemb, size_t size);`
- `void *realloc(void *ptr, size_t size);`
- `void free(void *ptr);`

ARRAY ALLOCATION

```
int n;  
int *list;  
...  
printf("How many numbers are you going to enter ?");  
scanf("%d", &n);  
list = (int *) malloc( n * sizeof(int) );  
if(list==NULL) {  
    printf("%s:%d>Can not allocate memory for the array...\n",__FILE__, __LINE__);  
    return -1;  
}
```

MATRIX ALLOCATION

```
int **mat;
int n,m;
printf("Please enter number of rows");scanf("%d", &n);
printf("Please enter number of columns");scanf("%d", &m);
mat = (int **) malloc( n * sizeof(int *) );
if(mat == NULL) {
    printf("%s:%d>Can not allocate memory for the array...\n",__FILE__, __LINE__);
    return -1;
}
for(i = 0; i < n; i++) {
    mat[i] = (int *)malloc(m * sizeof(int) );
}
```


EXAMPLE 1

- Write a simple program
 - ask number of elements in the array
 - allocate necessary space
 - ask for elements
 - sort the array