LECTURE NOTE 31

DYNAMIC MEMORY ALLOCATION

The memory allocation that we have done till now was static memory allocation.. So the memory that could be used by the program was fixed. So we couldnot allocate or deallocate memory during the execution of the program. It is not possible to predict how much memory will be needed by the program at run time. For example assume we have declared an array with size 20 elements, which is fixed. So if at run time values to be stored in array is less than 20 then wastage of memory occurs or our program may fail if more than 20 values are to be stored in to that array. To solve the above problems and allocate memory during runtime we are using dynamic memory allocation.

The following functions are used in dynamic memory allocation and are defined in <stdlib.h>

1. **malloc()**
   Declaration: *void *malloc(size_t size);*
   This function is used to allocate memory dynamically. The argument size specifies the number of bytes to be allocated. On success, malloc() returns a pointer to the first byte vof allocated memory. The returned pointer is of type void, which can be type cast to appropriate type of pointer. The memory allocated by malloc() contains garbage value
2. **calloc()**
   Declaration: *void *calloc(size_t n,size_t size);*
   This function is used to allocate multiple blocks of memory. The first argument specifies the number of blocks and the second one specifies the size of each block. The memory allocated by calloc() is initialized to zero.
3. **realloc()**
   Declaration: *void *realloc(void *ptr,size_t newsize);*
   The function realloc() is used to change the size of the memory block. It alters the size of the memory block without losing the old data. This function takes two arguments, first is a pointer to the block of memory that was previously allocated by mallloc() or calloc() and second one is the new size for that block.
4. **free();**
   Declaration: *void free(void *p);*
   This function is used to release the memory space allocated dynamically. Rhe memory released by free() is made available to the heap again and can be used for some other purpose. We should not try to free any memory location that was not allocated by malloc(), calloc() or realloc().

The following program illustrates Dynamic memory allocation.

```
#include<stdio.h>
#include<stdlib.h> void
main()
{ int *p,n,i;
 printf(“Enter the number of integers to be entered”);
scanf(“%d”,&n);
p=(int *)malloc(n*sizeof(int));    /* This is same as “(int *)calloc(n,sizeof(int))”*/
 /* If we write “(int *)malloc(sizeof(int))” then only 2 byte of memory will be allocated
dynamically*/
if(p==NULL)
{
printf(“Memory is not available”);
exit(1);
 }
 for(i=0;i<n;i++)
{
 printf(“Enter an integer”);
scanf(“%d”,p+i);
}
 for(i=0;i<n;i++)
printf(”%d\t”,*(p+i));
}
```