Chapter - 9 : Pointers

Dynamic Memory Allocation

- In general, there are two types of storages available for variables. They are
  1. Stack and
  2. Heap memory storages.
- The stack memory is permanent storage area where the ordinary variables can be stored. The heap memory is the free memory area available in the main memory of the computer which will be used for dynamic memory allocation.
- In C, there exists a set of built-in functions which are used to allocate memory dynamically to the derived data types like arrays, structures and unions.
- Some of the important functions are:
  1. malloc ()
  2. calloc () and
  3. free ()
- To use these functions, we need to include the **alloc.h** or **stdlib.h** header file.

Comparison between Static and Dynamic Memory Allocation

Static Memory Allocation	**Dynamic Memory Allocation**
1. This memory is allocated at compile time.	This memory is allocated at run time.
2. Memory allocation can not be modified while executing program.	Memory allocation can be changed while executing program.
3. Used in an array.	Used in a linked list.
4. It is fast and saves running time.	It is a bit slow.
5. It allocates memory from stack.	It allocates memory from heap.
6. Allocated memory stays till the end of program.	Memory can be allocated at any time and can be released at any time.
7. It is less efficient than a Dynamic allocation strategy.	It is more efficient than a Static allocation strategy.
8. Implementation is simple.	Implementation is complicated.
9. Example: int i; float j;	Example: p = malloc(sizeof(int));

malloc ( )

- This function allocates single block for memory.
- The malloc() function does not initialize the allocated memory by default. So it contains the garbage value by default.
- The general syntax of the malloc () function is
  pointer variable = (datatype *) malloc (no. of bytes to be allocated);

Example

```
int *x, n;
x=(int *) malloc(n * sizeof (int) );
```

"**x**" is the pointer variable of **int** data type and
"n" is an **int** type variable which can hold the value of the size of the array to be read-in.

The malloc() function allocates a single contiguous memory of "**n**" locations of **int** type for the pointer variable "**x**", such that "**x**" can become like an array of **"n" int** values, and the memory location address of the first element of the array is stored in "**x**".

Program – To find the sum of n elements entered by user. Use the dynamic memory to store the data and malloc () function.

```
#include<stdio.h>
#include<alloc.h>
#include<stdlib.h>
void main( )
{
        int n,i, *ptr, sum=0;
        printf("Enter number of Elements : ");
        scanf("%d", &n);

        ptr=(int*) malloc(n * sizeof(int)));

        if(ptr==NULL)
        {
         printf("Error ! Memory not allocated.");
         exit(0);
        }

        printf("Enter Elements of Array: "");
        for(i=0; i<n ; i++)
         {
                scanf("%d", &ptr[i]);
                sum=sum+ptr[i];
         }

         printf("Sum = %d", sum);
}
```

calloc( )

- This function allocates the multiple blocks for memory.
- The calloc() function initialize the allocated memory.
- The general syntax of the calloc () function is
  pointer variable = (datatype *) calloc (no. of location, size of each location);

Example

```
int *x, n;
x = (int *) calloc( n, sizeof( int ) );
```

"**x**" is the pointer variable of **int** data type and
"n" is an **int** type variable which can hold the value of the size of the array to be read-in.

The **calloc**( ) function allocates a number of blocks of memory in contiguous form and not as a single contiguous block of memory allocated by **malloc**( ) function.

Program – To find the sum of n elements entered by user. Use the dynamic memory to store the data and calloc () function.

```
#include<stdio.h>
#include<alloc.h>
#include<stdlib.h>
void main( )
{
        int n,i, *ptr, sum=0;
        printf("Enter number of Elements : ");
        scanf("%d", &n);

        ptr=(int*) calloc(n ,  sizeof(int)));

        if(ptr==NULL)
        {
         printf("Error ! Memory not allocated.");
         exit(0);
        }
        printf("Enter Elements of Array: "");
        for(i=0; i<n ; i++)
         {
                scanf("%d", &ptr[i]);
                sum=sum+ptr[i];
         }
         printf("Sum = %d", sum);
}
```

Comparison between malloc ( ) and calloc ( ) functions

	malloc ()	**calloc** ()
1.	It allocates single contiguous block of memory only.	It allocates number of blocks of memory in contiguous form.
2.	Memory allocation is not initialized, by default.	Memory allocation is initialized, by default.
3.	It is used to allocate memory for basic data types (like int, char, float and double).	It is used to allocate memory for derived data types (like arrays, structures and unions).

free( )

- Dynamically allocated memory with either calloc( ) or malloc( ) does not get return on its own.
- The programmer must use free( ) explicitly to release space.

Syntax of free( )

free(ptr);

This statement causes the space in memory pointer by ptr to be deallocated

Example

```
#include<stdio.h>
#include<alloc.h>
#include<stdlib.h>
void main( )
{
        int n,i, *ptr, sum=0;
        printf("Enter number of Elements : ");
        scanf("%d", &n);
        ptr=(int*) malloc(n * sizeof(int)));

        if(ptr==NULL)
        {
         printf("Error ! Memory not allocated.");
         exit(0);
        }

        free(ptr);

        if(ptr==NULL)
        {
          printf("Error ! Memory not allocated.");
          exit(0);
        }
}
```