C Dynamic Memory Allocation

Dynamically allocate memory in your C program using standard library functions:

- **malloc(),**
- **calloc()**
- **free()**
- **realloc().**

These functions are defined in the `<stdlib.h>` header file.

malloc()

The name "malloc" stands for memory allocation.

The `malloc()` function reserves a block of memory of the specified number of bytes. And, it returns a pointer of `void` which can be casted into pointers of any form.

Syntax of malloc()

```
ptr = (castType*) malloc(size);
```

Example

```
ptr = (float*) malloc(100 * sizeof(float));
```

The above statement allocates 400 bytes of memory. It's because the size of `float` is 4 bytes. And, the pointer `ptr` holds the address of the first byte in the allocated memory.

C calloc()

The name "calloc" stands for contiguous allocation.

The `malloc()` function allocates memory and leaves the memory uninitialized. Whereas, the `calloc()` function allocates memory and initializes all bits to zero.

Syntax of calloc()

```
ptr = (castType*)calloc(n, size);
```

Example:

`ptr = (float*) calloc(25, sizeof(float));`The above statement allocates contiguous space in memory for 25 elements of type `float`.

C free()

Dynamically allocated memory created with either `calloc()` or `malloc()` doesn't get freed on their own. You must explicitly use `free()` to release the space.

Syntax of free()

`free(ptr);`statement frees the space allocated in the memory pointed by `ptr`.

C realloc()

If the dynamically allocated memory is insufficient or more than required, you can change the size of previously allocated memory using the `realloc()` function.

Syntax of realloc()

```
ptr = realloc(ptr, x);
```

Here, `ptr` is reallocated with a new size `x`.

```
int main(){

    int *ptr, i , n1, n2;

    printf("Enter size: ");

    scanf("%d", &n1);


    ptr = (int*) malloc(n1 * sizeof(int));


    printf("Addresses of previously allocated memory: ");

    for(i = 0; i < n1; ++i)

        printf("%u\n",ptr + i);


    printf("\nEnter the new size: ");

    scanf("%d", &n2);
```

```
        // rellocating the memory
        ptr = realloc(ptr, n2 * sizeof(int));

        printf("Addresses of newly allocated memory: ");
        for(i = 0; i < n2; ++i)
                printf("%u\n", ptr + i);

        free(ptr);

        return 0;
}
```

When you run the program, the output will be:

```
Enter size: 2
Addresses of previously allocated memory:26855472
26855476

Enter the new size: 4
Addresses of newly allocated memory:26855472
26855476
26855480
26855484
```