Ch. 11 Dynamic Memory Allocation

In C is a language with some fixed rules of programming. For e.g. changing the size of an array is not allowed.

Dynamic Memory Allocation

Dynamic memory allocation is a way to allocate memory to memory to a data structure during the runtime. We can use DMA functions available in C to allocate and free memory during runtime.

Functions for DMA in C

1)

1) malloc()
2) calloc()
3) free()
4) realloc()

Malloc() function

Malloc stands for memory allocation. It takes number of bytes to be allocated as an input & returns a pointer of type void. The expression returns a NULL pointer if the memory cannot be allocated.

Syntax

```
ptr = (int *) malloc (30 * sizeof (int))
```

- (int *): Casting void pointer to int
- 30: Space for 30 ints
- sizeof (int): Returns the size of 1 int

Quiz: Write a program to create a dynamic array of ~~3~~ ~~ints~~ 5 integers using malloc

Code:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
int *ptr;
ptr = (int *) malloc (6 * sizeof(int));
for (int i = 0; i<6; i++) {
printf("Enter the value of %d element: \n", i);
scanf("%d", &ptr[i]);
}
```

```
for(int i=0; i<6; i++){
  printf("The value of %d element is: %d\n",
                    i, ptr[i]);
  }
return 0;
}
```

Output

Enter the value of 0 element : 1

1	2
2	3
3	4
4	5
5	6

The value of 0 element is : 1

1	2
2	3
3	4
4	5
5	6

Calloc() function

Calloc stands for continuous allocation. It initializes each memory block with a default value of 0.

Syntax:

```
ptr = (float *) calloc (30, sizeof(float));
```

allocates contiguous space in memory for 30 blocks of float

If space is not sufficient, memory allocation fails & a null pointer is returned

Quiz:

Code:
```
#include<stdio.h>
#include<stdlib.h>

int main(){
    int *ptr;
    int n;
    printf("How many ~~elements~~ integers do you
                    want to enter:\n");
    scanf("%d", &n);
    ptr=(int*)calloc(n, sizeof(int));

    for(int i=0;i<n;i++){
        printf("Enter the value of %d element:\n",i);
    scanf("%d", &ptr[i]);
    }
    for(int i=0; i<n; i++){
    printf("The value of %d element is:%d\n", i,
                    ptr[i]);
            }

    return 0;
}
```

How many integers do you want to enter
□ ≅ 2

Enter the value of 0 element : 1
" " " 1 " : 2

The value of 0 element is 1
" " " 1 " 2

Code:
```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int* ptr
    ptr=(int*) calloc(6, sizeof(int));
    for(int i=0; i<6; i++){
printf("The value of %d element is: %d
    \n", i, ptr[i]);
    }
return 0;
}
```

Output
The value of 0 element is: 0
" " " 1 " : —
2
3
4
5

Free() function

We can use free() function to deallocate the memory.
The memory allocated using calloc/malloc is not deallocated automatically.

Syntax

free(ptr); ⇒ Memory of ptr is released

Realloc() function

Sometimes the dynamically allocated memory is insufficient or more than required.

realloc is used to allocate memory of new size using the previous pointer and size.

Syntax:

ptr = realloc(ptr, newSize);

ptr = realloc(ptr, 3 * sizeof(int));
↳ ptr now points to the new block of memory capable of storing 3 integers

Ch.11 Practice Set

Q1 Write a program to dynamically create an array of size 6 capable of storing 6 integers & use that array to store 6 integers

Q1) Create an array dynamically capable of storing 5 integers. Now use realloc so that it can now store 10 integers

```
#include <stdio.h>
#include <stdlib.h>

int main(){
    int *ptr;
    ptr=(int*)malloc(5*sizeof(int));
    for(int i=0;i<5;i++){
    printf("Enter the value of %d element:\n",
                                    i);
    scanf("%d",&ptr[i]);
    }

    for(int i=0;i<5;i++){
    printf("The value of %d element is:%d\n",
                                    i,ptr[i]);
    }

    ptr=realloc(ptr,10*sizeof(int));
    for(int i=0;i<10;i++){
     printf("Enter the value of %d element:
                                    \n",i);
    scanf("%d",&ptr[i]);
    }
```

return 0;

}

Output

Enter the value of 0 element: 1

1

2

3

4

2

3

4

5

The value of 0 element is 1

2

3

4

5

Enter the value of 0 element:

1

9

The value of 0 element is

Q2: Create an array of multiplication table of 7 upto 10 (7X10=70). Use realloc to make it store 15 numbers (from 7X1 to 7X15).

Code:

```
#include <stdio.h>
#include <stdlib.h>

int main(){
int* ptr;
ptr=(int*)malloc(10*sizeof(int));
for(int i=0; i<10; i++){
  ptr[i]= 7*(i+1);
printf("The value of 7X%d=%d\n",
                    i+1, ptr[i]);
  }

ptr= realloc(ptr, 15*sizeof(int));
for(int i=0; i<15; i++){
    ptr[i]= 7*(i+1);
printf("The value of 7X%d=%d\n",
                    i+1, ptr[i]);
  }
return 0;
}

printf("\n After reallocating ---\n\n");
```

Output

The value of $7 \times 1 = 7$

$7 \times 2 = 14$

$7 \times 10 = 70$

After reallocating.....

The value of $7 \times 1 = 7$

$7 \times 2 = 14$

$7 \times 15 = 105$