CS118 Programming Fundamentals

LAB 11

Dynamic Memory Allocation

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Dynamic Memory Allocation

As you know, an array is a collection of a fixed number of values. Once the size of an array is declared, you cannot change it. Sometimes the size of the array you declared may be insufficient. To solve this issue, you can allocate memory manually during run-time. This is known as dynamic memory allocation in C programming.

To allocate memory dynamically, library functions are malloc(), calloc(), realloc() and free() are used. These functions are defined in the <stdlib.h> header file.

Function	Purpose
malloc	Allocates the memory of requested size and returns the pointer to the first byte of allocated space.
calloc	Allocates the space for elements of an array. Initializes the elements to zero and returns a pointer to the memory.
realloc	It is used to modify the size of previously allocated memory space.
Free	Frees or empties the previously allocated memory space.

C 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:

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

Example:

```
ptr = (int*) 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. The expression results in a NULL pointer if the memory cannot be allocated.

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:

```
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:

free(ptr);

This 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:

ptr = realloc(ptr, x);

Here, ptr is reallocated with a new size x.

Example 1: malloc() and free()

```
// Program to calculate the sum of n numbers entered by the
user #include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, i, *ptr, sum = 0;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    ptr = (int*) malloc(n * sizeof(int));
    // if memory cannot be
    allocated if(ptr == NULL)
    {
        printf("Error! memory not allocated.");
        exit(0);
    }
    printf("Enter elements: ");
    for(i = 0; i < n; ++i)
    {
        scanf("%d", ptr + i);
        sum += *(ptr + i);
    }
    printf("Sum = %d", sum);

    // deallocating the memory
    free(ptr);

    return 0;
}
```

Example 2: calloc() and free()

```
// Program to calculate the sum of n numbers entered by the
user #include <stdio.h>
#include <stdlib.h>
int 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: ");
    for(i = 0; i < n; ++i)
    {
        scanf("%d", ptr + i);
        sum += *(ptr + i);
    }
    printf("Sum = %d", sum);
    free(ptr);
    return 0;}
```

Example 3: realloc()

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

calloc vs. malloc:

The calloc function is generally more suitable and efficient than that of the malloc function. While both the functions are used to allocate memory space, calloc can allocate multiple blocks at a single time. You don't have to request for a memory block every time. The calloc function is used in complex data structures which require larger memory space.

The memory block allocated by a calloc function is always initialized to zero while in malloc it always contains a garbage value.

LAB TASK

Question # 01:

Write a function that print int, float and char values using void pointers but you have to declare a single void pointer variable. And its up to user which type of value it enters.

Question # 02:

Write a C program to create memory for int, char and float variable at run time by using dynamic memory allocation.

Question # 03:

Write a program in C to find the smallest element using Dynamic Memory Allocation.
Input total number of elements (1 to 50): 6
Number 1: 4
Number 2: 7
Number 3: 1
Number 4: 19
Number 5: 2
Number 6: 8
Output: The smallest number is 1

Question # 04:

Suppose your Programming Fundamental teacher provided you the midterm marks of your section. Your task is to find the highest marks and 2^{nd} highest marks from the given list by C programming. Also find average marks of the class. Write a C program by using dynamic memory allocation.

Question # 05:

Suppose you want to put a toy in a box, but you only have an approximate idea of its size. For that, you would require a box whose size is equal to the approximate size of the toy. Write a program that satisfies the above statement.

Question # 06:

Write a C program in which you have to create memory for text string at run time using `malloc()` function, text string will be inputted by the user and displayed. Using `free()` function we will release the occupied memory.

Question # 07:

Write a C program in which you have to take three variables from user of type int, char and float and by using a void pointer, pointing the addresses of all these variables and print values of all variables along with their addresses .