

**Lab No. 1 Date:2081/10/09 Title: Write a program to demonstrate dynamic memory allocation in C.**

Dynamic memory allocation using malloc(), calloc(), free(), and realloc() is essential for efficient memory management in C.

Therefore, Dynamic Memory Allocation can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime.

C provides some functions to achieve these tasks. There are 4 library functions provided by C defined under <stdlib.h> header file to facilitate dynamic memory allocation in C programming. They are:

1. malloc()

Syntax: ptr = (cast-type\*) malloc(byte-size)

2. calloc()

Syntax: ptr = (cast-type\*)calloc(n, element-size);

3. free()

Syntax: free(ptr);

4. realloc()

Syntax: ptr = realloc(ptr, newSize);

Complier: DEV C++

Language: C

Source Code:

*//Using malloc*

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int n, \*ptr, i, m;

    printf("DMA \nCompiled by Jonash\n");

    printf("Enter the size of n: ");

    scanf("%d", &n);

    // Allocate memory

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

    if (ptr == NULL)

    {

        printf("Memory allocation failed\n");

        return 1;

    }

    // Print garbage values

    printf("Garbage values in the allocated memory:\n");

    for (i = 0; i < n; i++)

    {

        printf("%d\n", \*(ptr + i));

    }

    printf("Now initializing\n");

    // Initialize the values by asking the user

    printf("Enter the value of the elements:\n");

    for (i = 0; i < n; i++)

    {

        scanf("%d", ptr + i);

    }

    // Print the initialized values

    printf("The values of the elements are:\n");

    for (i = 0; i < n; i++)

    {

        printf("%d ", \*(ptr + i));

    }

    printf("\n");

    // Reallocate memory

    printf("Enter the new size (i.e. m) for reallocating the size of n: ");

    scanf("%d", &m);

    ptr = (int \*)realloc(ptr, m \* sizeof(int));

    if (ptr == NULL)

    {

        printf("Memory reallocation failed\n");

        return 1;

    }

    // Print garbage values after reallocation

    printf("Garbage values after the reallocated memory:\n");

    for (i = 0; i < m; i++)

    {

        // printf("%d\n", \*(ptr + i));

        printf("%d\n", ptr[i]);

    }

    printf("Now initializing the garbage values\n");

    // Initialize the values by asking the user

    printf("Enter the value of for the new elements:\n");

    for (i = 0; i < m - n; i++)

    {

        scanf("%d", ptr + i + n);

    }

    // Print the initialized values

    printf("The values of all elements are:\n");

    for (i = 0; i < m; i++)

    {

        printf("%d ", \*(ptr + i));

    }

    printf("\n");

    // Free the allocated memory

    free(ptr);

    return 0;

}

    // Initialize the values by asking the user

    printf("Enter the value of for the new elements:\n");

    for (i = 0; i < m - n; i++)

    {

        scanf("%d", ptr + i + n);

    }

    // Print the initialized values

    printf("The values of all elements are:\n");

    for (i = 0; i < m; i++)

    {

        printf("%d ", \*(ptr + i));

    }

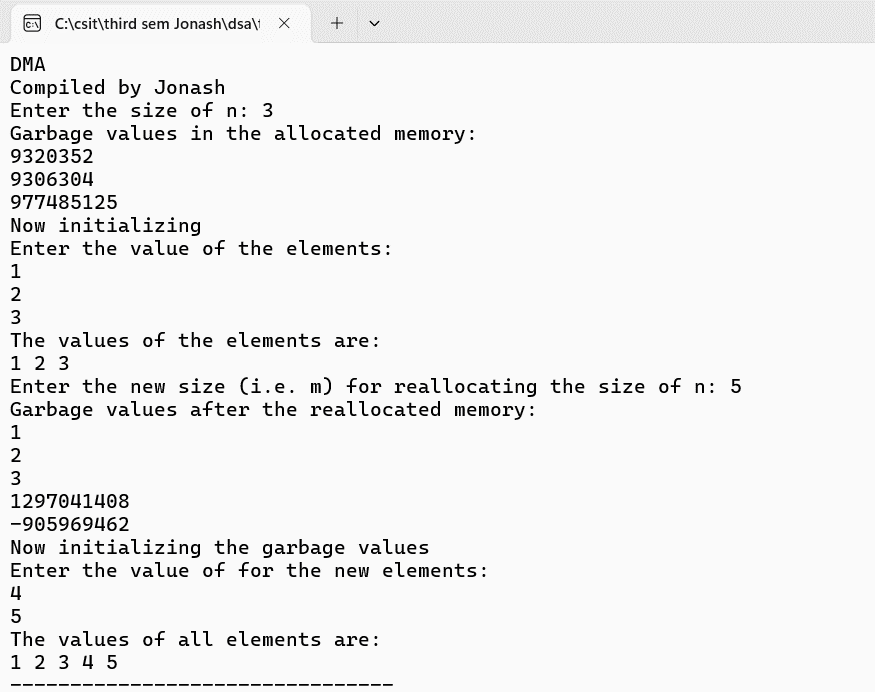
    // Free the allocated memory

    free(ptr);

    return 0;

}

**Output:**



Source Code:

*// Using calloc*

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int n, \*ptr, i, m;

    printf("DMA using Calloc \nCompiled by Jonash\n");

    printf("Enter the size of n: ");

    scanf("%d", &n);

    // Allocate memory

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

    if (ptr == NULL)

    {

        printf("Memory allocation failed\n");

        return 1;

    }

    // Print garbage values

    printf("Garbage values in the allocated memory:\n");

    for (i = 0; i < n; i++)

    {

        printf("%d\n", \*(ptr + i));

    }

    printf("Now initializing\n");

    // Initialize the values by asking the user

    printf("Enter the value of the elements:\n");

    for (i = 0; i < n; i++)

    {

        scanf("%d", ptr + i);

    }

    // Print the initialized values

    printf("The values of the elements are:\n");

    for (i = 0; i < n; i++)

    {

        printf("%d ", \*(ptr + i));

    }

    printf("\n");

    // Reallocate memory

    printf("Enter the new size (i.e. m) for reallocating the size of n: ");

    scanf("%d", &m);

    ptr = (int \*)realloc(ptr, m \* sizeof(int));

    if (ptr == NULL)

    {

        printf("Memory reallocation failed\n");

        return 1;

    }

    // Print garbage values after reallocation

    printf("Garbage values after the reallocated memory:\n");

    for (i = 0; i < m; i++)

    {

        // printf("%d\n", \*(ptr + i));

        printf("%d\n", ptr[i]);

    }

    printf("Now initializing the garbage values\n");

    // Initialize the values by asking the user

    printf("Enter the value of for the new elements:\n");

    for (i = 0; i < m - n; i++)

    {

        scanf("%d", ptr + i + n);

    }

    // Print the initialized values

    printf("The values of all elements are:\n");

    for (i = 0; i < m; i++)

    {

        printf("%d ", \*(ptr + i));

    }

    printf("\n");

    // Free the allocated memory

    free(ptr);

    return 0;

}

// Initialize the values by asking the user

    printf("Enter the value of for the new elements:\n");

    for (i = 0; i < m - n; i++)

    {

        scanf("%d", ptr + i + n);

    }

    // Print the initialized values

    printf("The values of all elements are:\n");

    for (i = 0; i < m; i++)

    {

        printf("%d ", \*(ptr + i));

    }

    printf("\n");

    // Free the allocated memory

    free(ptr);

    return 0;

}

**Output:**

