Dynamic memory Allocations and Dynamic arrays

PRESENTED BY: SATYA PRAKASH PATEL
EMAIL: SATYAPATEL.IND@GMAIL.COM

Dynamic memory Allocation

There are 4 library functions defined under <stdlib.h> makes dynamic memory allocation in C programming.

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

malloc()

"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 type void which can be casted into pointer of any form.

Syntax of malloc()

ptr = (cast-type*) malloc(byte-size)

Example

ptr = (int*) malloc(100 * sizeof(int));

calloc()

The name "calloc" stands for contiguous allocation.

The malloc() function allocates a single block of memory. Whereas, calloc() allocates multiple blocks of memory and initializes them to zero.

Syntax of calloc()

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

Example

ptr = (float*) calloc(25, sizeof(float));

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);

This statement frees the space allocated in the memory pointed by ptr.

realloc()

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

Syntax

ptr = realloc(ptr, x);

Dynamic Array

Sometimes, the size of array we declared may be insufficient. To solve this issue, we can allocate memory manually during run-time.

1D Dynamic Array

```
int *arr = (int *)malloc(n * sizeof(int));
```

```
#include<stdio.h>
int main()
{

    int i,n;
    scanf("%d", &n);

    int *arr;
    arr= (int*)malloc(n*sizeof(int));
    for(i=0;i<n;i++)
    { arr[i]=i;
    printf("\na[%d]= %d", i,arr[i]);

    }
return 0;
}
```

2D Dynamic Array

```
int *arr[r];

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

        arr[i] = (int *)malloc(c * sizeof(int));
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int r = 3, c = 4, i, j, count;
    int *arr[r];
    for (i=0; i<r; i++)
        arr[i] = (int *)malloc(c * sizeof(int));     // Note that arr[i][j] is same as *(*(arr+i)+j)
    count = 0;
    for (i = 0; i <  r; i++)
      for (j = 0; j < c; j++)
        arr[i][j] = ++count; // Or *(*(arr+i)+j) = ++count

    for (i = 0; i <  r; i++)
     {
      for (j = 0; j < c; j++)
        printf("%d\t", arr[i][j]);
        printf("\n");
    }
   return 0;
}
```

Question 1 Predict output of following program

```
#include<stdio.h>

int main()

{

   int i;

   char arr[5] = {65};

   for (i = 0; i < 5; i++)

      printf("%c ", arr[i]);

   return 0;

}
```

(A) A followed by four garbage values:
(B) A 0 0 0 0
(C) 65 1 1 1 1
(D)A

```
#include<stdio.h>
int main()
{
    int i;
    int arr[5]={010, 020,030,040,050};
    int arr1[5]={0x10, 0x20,0x30,0x40,0x50};
     int arr2[5]={10, 20,30,40,50};
    for (i = 0; i < 5; i++)
        printf("%d ", arr[i]);printf("\n");

    for (i = 0; i < 5; i++)
        printf("%d ", arr1[i]);printf("\n");

    for (i = 0; i < 5; i++)
        printf("%o ", arr[i]);printf("\n");

    for (i = 0; i < 5; i++)
        printf("%X ", arr2[i]);

    return 0;
}
```

Question 2 Predict output of following program

A 8 16 24 32 40

16 32 48 64 80

10 20 30 40 50

A 14 1E 28 32

B 10 20 30 40 50
10 20 30 40 50
10 20 30 40 50
A 14 1E 28 32

C 8 16 24 32 64
10 20 30 40 50
10 20 30 40 50
A 14 1E 28 32

D 10 20 30 40 50
10 20 30 40 50
10 20 30 40 50
a 14 1e 28 32

Question 3 Predict output of following program

```
#include<stdio.h>
int main()
{
   int i;
   int arr[5]={010, 020,030,040,050};
   int arr1[5]={0x10, 0x20,0x30,0x40,0x50};
    int arr2[5]={10, 20,30,40,50};
   for (i = 0; i < 5; i++)
      printf("%d ", arr[i]|arr1[i]);
   return 0;
}
```

A. 10 20 30 40 50
B. 24 48 56 96 120
C. 16 32 48 64 80
D. None of the above

Question 4 Predict output of following program

```
#include<stdio.h>
int main()
{
   int i;
   int arr[5]={010, 020,030,040,050};
   int arr1[5]={0x10, 0x20,0x30,0x40,0x50};

   for (i = 0; i < 5; i++)
      printf("%o ", arr[i]+arr1[i]);
   return 0;
}
```

A. 10 20 30 40 50
B. 24 48 56 96 120
C. 16 32 48 64 80
D. 30 60 110 140 170

Queries and Feedback