C Programming Lecture 14

Monday, 24 June 2024 8:18 PM

Dynamic memory allocation in C

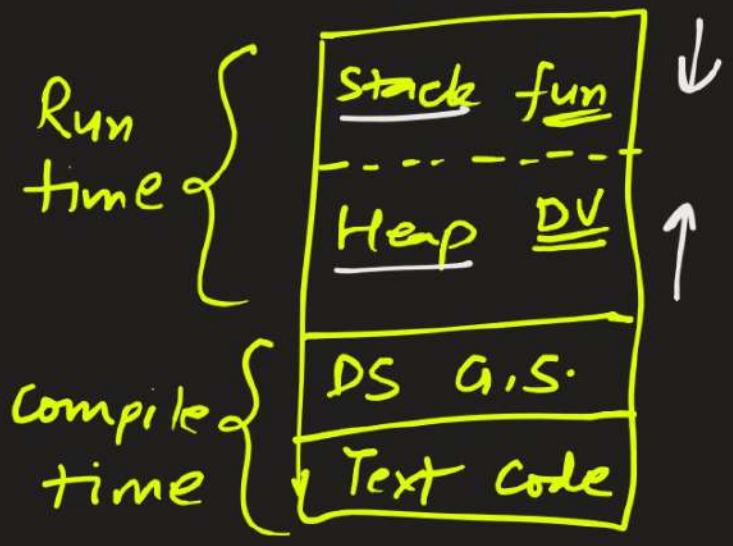
The concept of **dynamic memory allocation in c language** *enables the C programmer to allocate memory at runtime.* Dynamic memory allocation in c language is possible by 4 functions of `stdlib.h` header file.

include < stdlib.h >

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

static memory allocation	dynamic memory allocation
memory is allocated at compile time.	memory is allocated at run time.
memory can't be increased while executing program.	memory can be increased while executing program.
used in array.	used in linked list.

malloc()	allocates single block of requested memory.
calloc()	allocates multiple block of requested memory.
realloc()	reallocates the memory occupied by malloc() or calloc() functions.
free()	frees the dynamically allocated memory.



n

scanf ("%d", &n);

int a[n]; ✗

Dynamic memory

malloc (40);

40B

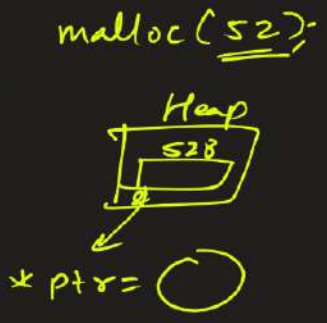
calloc (10, 4);

number

size of

return NULL if memory is not avaible.

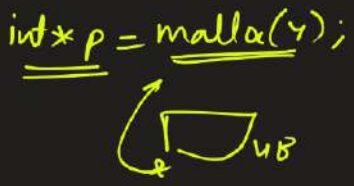
malloc() function in C



- The malloc() function allocates single block of requested memory.
- It doesn't initialize memory at execution time, so it has garbage value initially.
- It returns NULL if memory is not sufficient.

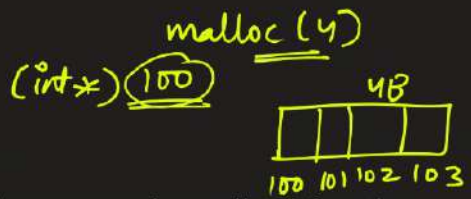
```
ptr = (cast-type*)malloc(byte-size)
```

calloc() function in C



- The calloc() function allocates multiple block of requested memory.
- It initially initialize all bytes to zero.
- It returns NULL if memory is not sufficient.

```
ptr = (cast-type*)calloc(number, byte-size)
```



realloc() function in C

- If memory is not sufficient for malloc() or calloc(), you can reallocate the memory by realloc() function.
- It changes the memory size of dynamically alloted memory.

```
ptr = realloc(ptr, new-size)
```

free() function in C

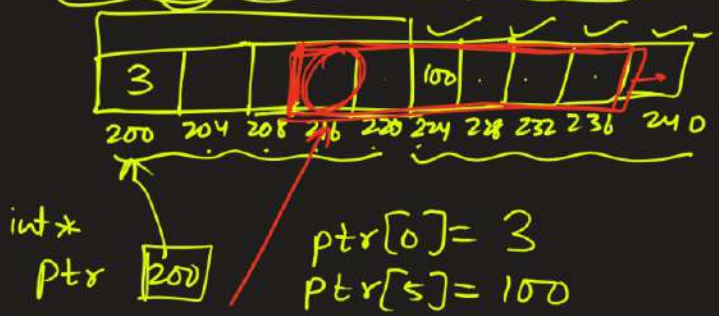
- The memory occupied by malloc() or calloc() functions must be released by calling free() function.
- Dynamically created variables will consume memory until program exit if not freed using free() function.

```
free(ptr)
```

```
int *ptr = malloc (4);
if (ptr == NULL)
```

```
int* p = (int*) malloc (4)
*p = 5;
```


```
int *ptr = (int *) calloc (10, 4)
```



ptr[0] = 3
ptr[5] = 100

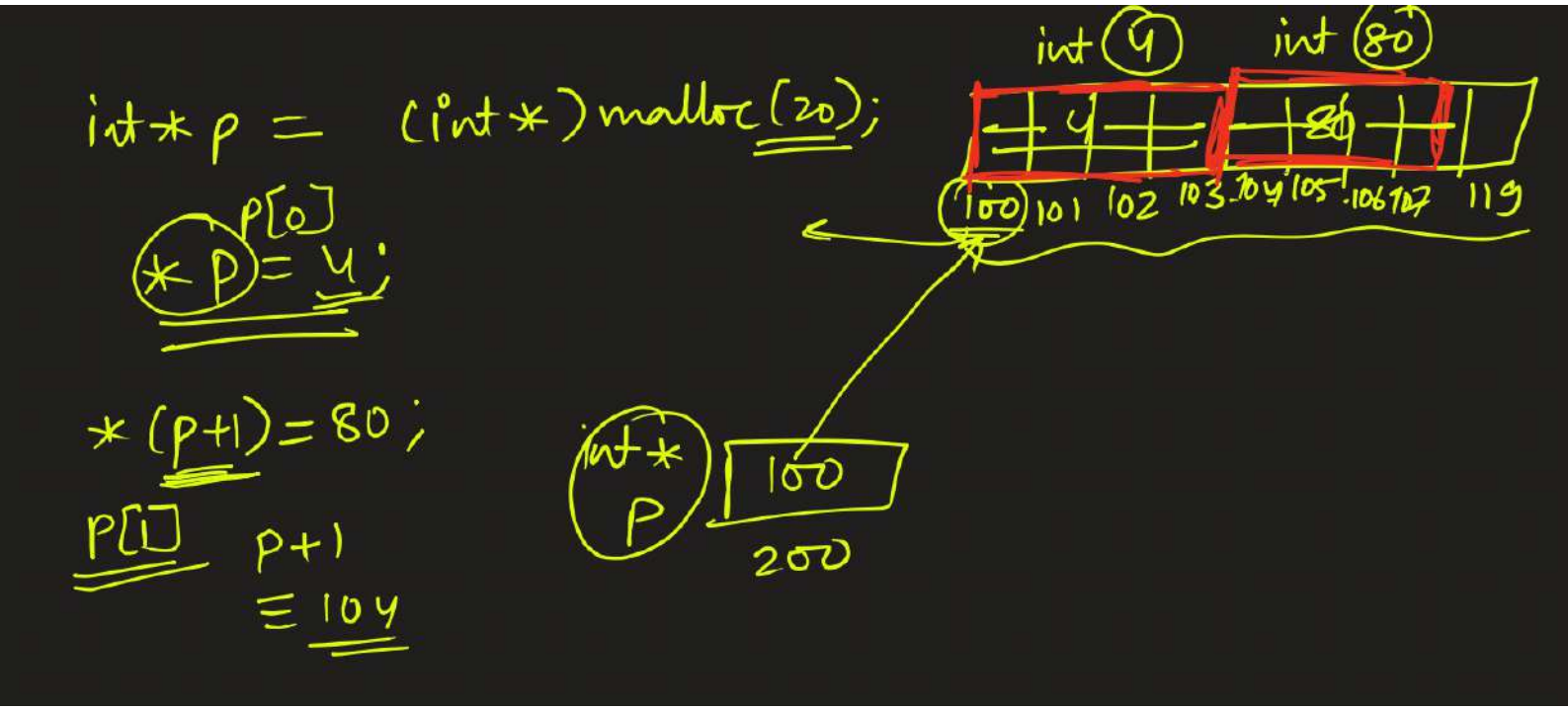
```
ptr = realloc ( ptr, 20);
```

```
x = 5;          sizeof (int) = 4B
malloc ( x * sizeof (int));
≡ malloc(20);
```

```
calloc ( 5, sizeof (int))
(floats)
5 * 4 = 20B
```

Example

```c
#include<stdio.h>
#include<stdlib.h>
void main(){
    int x = 10/3;
    int* mp = (int*)malloc(x*sizeof(int));  //memory allocated using malloc
    int* cp = (int*)calloc(x, sizeof(int));  //memory allocated using calloc
    mp[0] = 1, mp[1] = 2, mp[2] = 3;
    cp[0] = 4, cp[1] = 5, cp[2] = 6;
    printf("%d, %d, %d, Address=%p \n", mp[0], mp[1], mp[2], mp);
    printf("%d, %d, %d, Address=%p \n", cp[0], cp[1], cp[2], cp);
    mp = realloc(mp, sizeof(int));
    free(mp);
    printf("%d, %d, %d, Address=%p \n", mp[0], mp[1], mp[2], mp);
}
```


int* p = (int*) malloc(20);
p[0]
*p = 4;
*(p+1) = 80;
p[1]
p+1
≡ 104
int 4
int 80
100 101 102 103 104 105 106 107 119
int*
p
100
200