Size of dynamically allocated memory can be changed by using realloc().

```
void *realloc(void *ptr, size_t size);
```

realloc deallocates the old object pointed to by ptr and returns a pointer to a new object that has the size specified by size. The contents of the new object is identical to that of the old object prior to deallocation, up to the lesser of the new and old sizes. Any bytes in the new object beyond the size of the old object have indeterminate values.

The point to note is that **realloc() should only be used for dynamically allocated memory**. If the memory is not dynamically allocated, then behavior is undefined.

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int arr[2], i;
   int *ptr = arr;
   int *ptr_new;

   arr[0] = 10;
   arr[1] = 20;

   // incorrect use of new_ptr: undefined behaviour
   ptr_new = (int *)realloc(ptr, sizeof(int)*3);
   *(ptr_new + 2) = 30;

   for(i = 0; i < 3; i++)
     printf("%d ", *(ptr_new + i));

   getchar();
   return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int *ptr = (int *)malloc(sizeof(int)*2);
   int i;
   int *ptr_new;

   *ptr = 10;
   *(ptr + 1) = 20;

   ptr_new = (int *)realloc(ptr, sizeof(int)*3);
   *(ptr_new + 2) = 30;
   for(i = 0; i < 3; i++)
       printf("%d ", *(ptr_new + i));

   getchar();
   return 0;
}
```

Thank you for watching!

Please leave us your comments.