Mark as done

Two more allocation methods

In addition to the ***malloc*** function we've used so far, there're two more memory allocation functions it's worth knowing about 😊

calloc

Defined in header `<stdlib.h>`

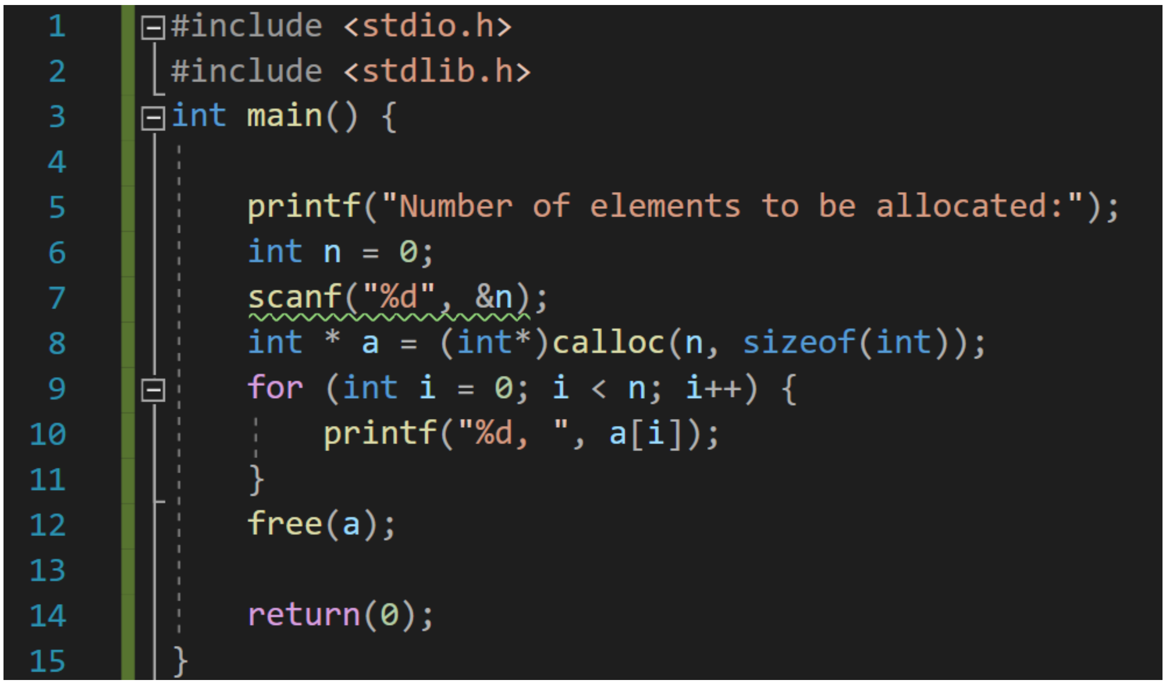
`void* calloc (size_t num, size_t size);`

Allocate and zero-initialize array

Allocates a block of memory for an array of num elements, each of them size bytes long, and initializes all its bits to zero.

The effective result is the allocation of a zero-initialized memory block of `(num*size)` bytes.

Example:



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

    printf("Number of elements to be allocated:");
    int n = 0;
    scanf("%d", &n);
    int * a = (int*)calloc(n, sizeof(int));
    for (int i = 0; i < n; i++) {
        printf("%d, ", a[i]);
    }
    free(a);

    return(0);
}
```

Output:

```
Number of elements to be allocated:10
0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

?

realloc

Defined in header `<stdlib.h>`

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

Reallocates the given area of memory. It must be previously allocated by malloc(), calloc() or `realloc()` and not yet freed with a call to free or `realloc`. Otherwise, the results are undefined.

The reallocation is done by either:

> a) expanding or contracting the existing area pointed to by `ptr`, if possible. The contents of the area remain unchanged up to the lesser of the new and old sizes. If the area is expanded, the contents of the new part of the array are undefined.
>
> b) allocating a new memory block of size `new_size` bytes, copying memory area with size equal the lesser of the new and the old sizes, and freeing the old block.

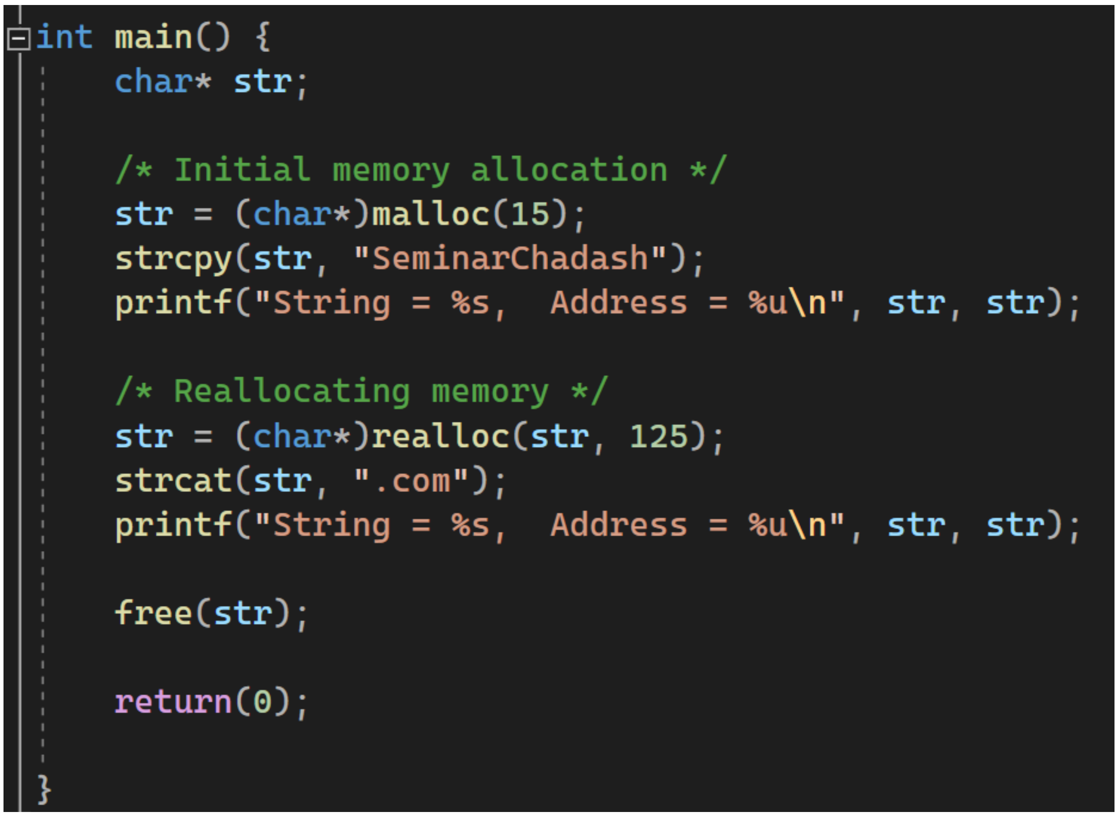
If there is not enough memory, the old memory block is not freed and null pointer is returned.

Parameters

ptr - pointer to the memory area to be reallocated

new_size - new size of the array in bytes

Example:



```c
int main() {
    char* str;

    /* Initial memory allocation */
    str = (char*)malloc(15);
    strcpy(str, "SeminarChadash");
    printf("String = %s,  Address = %u\n", str, str);

    /* Reallocating memory */
    str = (char*)realloc(str, 125);
    strcat(str, ".com");
    printf("String = %s,  Address = %u\n", str, str);

    free(str);

    return(0);

}
```

Output:

```
String = SeminarChadash, Address = 16873792
String = SeminarChadash.com, Address = 16907552
```

Last modified: Tuesday, 6 February 2024, 12:47 PM