

Dynamic Memory

```
Song songs[100];
```

... works fine. Until 100... Room for one more?

Increase/decrease such memory dynamically?

Simply no in C

Unless you use dynamic memory.

Dynamic memory

calloc and malloc - allocates memory

realloc - increases/decreases memory

free - returns memory back to OS/process

NAME

malloc, free, calloc, realloc - allocate and free dynamic memory

SYNOPSIS

```
#include <stdlib.h>
```

```
void *malloc(size_t size);
```

```
void free(void *ptr);
```

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

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


DESCRIPTION

The `malloc()` function allocates `size` bytes and returns a pointer to the allocated memory. The memory is not initialized. If `size` is 0, then `malloc()` returns either `NULL`, or a unique pointer value that can later be successfully passed to `free()`.

DESCRIPTION

The `calloc()` function allocates memory for an array of `nmemb` elements of size `bytes` each and returns a pointer to the allocated memory. The memory is set to zero. If `nmemb` or `size` is 0, then `calloc()` returns either `NULL`, or a unique pointer value that can later be successfully passed to `free()`.

DESCRIPTION

The `free()` function frees the memory space pointed to by `ptr`, which must have been returned by a previous call to `malloc()`, `calloc()`, or `realloc()`. Otherwise, or if `free(ptr)` has already been called before, undefined behavior occurs. If `ptr` is `NULL`, no operation is performed.

DESCRIPTION

The `realloc()` function changes the size of the memory block pointed to by `ptr` to `size` bytes. The contents will be unchanged in the range from the start of the region up to the minimum of the old and new sizes. If the new size is larger than the old size, the added memory will not be initialized. If `ptr` is `NULL`, then the call is equivalent to `malloc(size)`, for all values of `size`; if `size` is equal to zero, and `ptr` is not `NULL`, then the call is equivalent to `free(ptr)`. Unless `ptr` is `NULL`, it must have been returned by an earlier call to `malloc()`, `calloc()` or `realloc()`. If the area pointed to was moved, a `free(ptr)` is done.