EDUCBA
(https://www.educba.com/software-development/)

← (https://www.educba.com/typedef-in-c/)

→ (https://www.educba.com/linked-list-in-c/)

Introduction to Memory Allocation in C

Memory allocations, in general, mean where computer programs and services are execu
reserve partially or complete space or virtual memory of a computer, this process is known as memory allocation. This process is hardware operation and is achieved by memory

(https://www.educba.com/software-development/)

memory is allocated at compile time.

How does Memory Allocation work in C?

In C language, static and dynamic memory allocation is also known as stack memory and heap memory which are allocated during compile time and run time, respectively.

1. Static Memory Allocation

As we discussed static memory allocation is the allocation of memory for the data variables when the computer programs start. This type of allocation is applied to only global variables, file scope variables and also to those variables that are declared as static. This type of allocation is having a drawback when you are allocating memory we should know the exact memory before allocating as this process allocates fixed memory and cannot be changed after allocating.

1. There are a few features of static memory allocation. They are: this type of allocation allocates variables permanently; hence the memory in this type of allocation cannot be reused and is, therefore, less efficient. This allocation uses the stack for implementing the allocation process.

Let us see an example below:

Code:

```
void play
```

(https://www.educba.com/software-development/)

```
ı
int y;
int c[10];
return 1;
}
```

Explanation: In the above program, the variables x, y, and care statically allocated so the memory is strictly allocated at compile time for the variable data. Note that the deletion of memory is necessary when the variables are not in use because it will lead to memory leakage. Therefore in static memory allocation, it automatically frees the memory based on the scope of the variable which means as soon as the variable's cope is over the memory gets re

2. A variable can internally or externally be declared as static in which its value persists until the end of the program, where this can be done using the keyword static before the variable declaration. There can be internal or external static variables that are declared inside or outside the function.

Let us see an example:

```
#include<stdio.h>
void stat(void);
int main()
{
int i;
for(i=1; i<=3 ; i++)
stat();
return 1;
}
void stat(void)
{
static int n = 0;
n = n+1;
printf("n = %d""\n", n);
}
```


Output:

```
n = 1
n = 2
```

(https://www.educba.com/software-development/)

during program execution. Dynamic memory allocation provides different functions in the C programming language. They are: malloc(), calloc(), realloc(), free(). Let us see in detail.

1. malloc()

This method allocates the space in the memory during execution but will not initialize the memory allocation during execution as it carries garbage values and if it cannot allocate requested memory then it returns a null pointer.

Syntax:

```
(CastType*) malloc(size);
```

Code:

```
mptr = (int*) malloc(100 * sizeof (int));
```

In the above example, the statement allocates 200 bytes of memory because the int size in C is 2 bytes and the variable mptr pointer holds the address of the first byte in the memory.

2. calloc()

This is also known as contiguous allocation. As in malloc() will not initialize any memory bit. But in calloc() it allocates the memory along with initializing the bits to zero.

Syntax:


```
(CastType*) calloc(n, size)
```

Code:

(https://www.educba.com/software-development/)

elements of data type “int”.

3. free()

As discussed above the memory space should be freed or released if not in use. In Dynamic memory allocation, malloc() and calloc() function only allocates memory but cannot free the memory on their own so this is done using free() method explicitly to release the memory that is not in use to avoid memory leakage.

Syntax:

```
free (pointer_variable);
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *p, *p1;
int x, i;
x = 5;
printf("Enter number of elements to allocate in memory : %d\n",
x);
p = (int*)malloc(x * sizeof(int));
p1 = (int*)calloc(x, sizeof(int));
if (p == NULL || p1 == NULL) {
```

(https://www.educba.com/software-development/)

```
printf("Memory has been successfully allocated using malloc.\n");
free(p);
printf("Malloc Memory has been successfully freed or released.\n");
printf("\nMemory has been successfully allocated using calloc.\n");
free(p1);
printf("Calloc Memory has been successfully freed or released.\n");
}
return 0;
}
```

Output:

```
Enter number of elements to allocate in memory : 5
Memory has been successfully allocated using malloc.
Malloc Memory has been successfully freed or released.

Memory has been successfully allocated using calloc.
Calloc Memory has been successfully freed or released.
```

4. realloc()

As the name suggests, in dynamic memory allocation if suppose a user wants to allocat memory which means more memory than specified or required by the program then we can use this realloc() function to alter the size of memory that was allocated previously.

(https://www.educba.com/software-development/)

Code:

Suppose if we want to change the size of memory from 200 bytes to 600 bytes. Let us how it can be done using realloc().

```
char *rptr;
rptr = malloc(200);
rptr = realloc(rptr, 600);
```

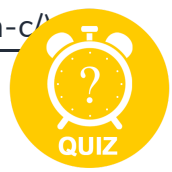
Conclusion

Memory allocation in C programming language is simple using static memory allocation which allocates memory during compile time or we can say before the program execution and it also has another type known as dynamic memory allocation which allocates memory during run time or allocating memory during program execution which uses 4 different functions such as malloc(), calloc(), free() and realloc(). There are different pros and cons of both methods.

Recommended Articles

This is a guide to Memory Allocation in C. Here we discuss static and dynamic memory allocation in C language with functions. You can also go through our other related articles to learn more –

1. Multidimensional Array in C (https://www.educba.com/multidimensional-array-in-c/)
2. Types of Memory in Java (https://www.educba.com/types-of-memory-in-java/)
3. C Literals (https://www.educba.com/c-literals/)
4. What is Heap Memory? (https://www.educba.com/what-is-heap-memory/)



About Us

(https://www.educba.com/software-development/)

source=footer)

Privacy Policy (https://www.educba.com/privacy-policy/?source=footer)

Apps

iPhone & iPad (https://itunes.apple.com/in/app/educba-learning-app/id1341654580?mt=8)

Android (https://play.google.com/store/apps/details?id=com.educba.www)

Resources

Free Courses (https://www.educba.com/software-development/free-courses/?source=footer)

Java Tutorials (https://www.educba.com/software-development/software-development-tutorials/java-tutorial/?source=footer)

Python Tutorials (https://www.educba.com/software-development/software-development-tutorials/python-tutorial/?source=footer)

All Tutorials (https://www.educba.com/software-development/software-development-tutorials/?source=footer)

Certification Courses

All Courses (https://www.educba.com/software-development/courses/?source=footer)

Software Development Course - All in One Bundle (https://www.educba.com/software-development/courses/software-development-course/?source=footer)

Become a Python Developer (https://www.educba.com/software-development/courses/python-certification-course/?source=footer)

Java Course (https://www.educba.com/software-development/courses/java-course/?source=footer)

(https://www.educba.com/software-development/)

VB.NET Course (https://www.educba.com/software-development/courses/vb-net-course/?source=footer)

PHP Course (https://www.educba.com/software-development/courses/php-course/?source=footer)