new and delete Operators in C++ For Dynamic Memory

Difficulty Level : Easy • Last Updated : 18 Oct, 2022

Dynamic memory allocation in C/C++ refers to performing memory allocation manually by a programmer. Dynamically allocated memory is allocated on **Heap,** and non-static and local variables get memory allocated on **Stack** (Refer to Memory Layout C Programs for details).

What are applications?

- One use of dynamically allocated memory is to allocate memory of variable size, which is not possible with compiler allocated memory except for variable-length arrays.
- The most important use is the flexibility provided to programmers. We are free to allocate and deallocate memory whenever we need it and whenever we don't need it anymore. There are many cases where this flexibility helps. Examples of such cases are Linked List, Tree, etc.

How is it different from memory allocated to normal variables?

For normal variables like "int a", "char str[10]", etc, memory is automatically allocated and deallocated. For dynamically allocated memory like "int *p = new int[10]", it is the programmer's responsibility to deallocate memory when no longer needed. If the programmer doesn't deallocate memory, it causes a memory leak (memory is not deallocated until the program terminates).

C uses the malloc() and calloc() function to allocate memory dynamically at run time and uses a free() function to free dynamically allocated memory. C++ supports these functions and also has two operators **new** and **delete,** that perform the task of allocating and freeing the memory in a better and easier way.

new operator

The new operator denotes a request for memory allocation on the Free Store. If sufficient memory is available, a new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.

Syntax to use new operator

```
pointer-variable = new data-type;
```

Here, the pointer variable is the pointer of type data-type. Data type could be any built-in data type including array or any user-defined data type including structure and class.
Example:

See a successful future with C++
A COMPLETE BEGINNER TO ADVANCED C++ COURSE
25+ Hours Of Content
150+ Curated Problems
24x7 Doubt Support
Explore Course

```
// Pointer initialized with NULL
// Then request memory for the variable
int *p = NULL;
p = new int;
```

```
// Combine declaration of pointer
// and their assignment
int *p = new int;
```

Initialize memory: We can also initialize the memory for built-in data types using a new operator. For custom data types, a constructor is required (with the data type as input) for initializing the value. Here's an example of the initialization of both data types :

```
pointer-variable = new data-type(value);
```

Example:

C++

```
int* p = new int(25);
float* q = new float(75.25);

// Custom data type
struct cust
{
    int p;
    cust(int q) : p(q) {}
    cust() = default;
    //cust& operator=(const cust& that) = default;
};

int main()
{
    // Works fine, doesn’t require constructor
    cust* var1 = new cust;

    //OR

    // Works fine, doesn’t require constructor
    var1 = new cust();

    // Notice error if you comment this line
    cust* var = new cust(25);
    return 0;
}
```

tart Your Coding Journey Now!

Allocate a block of memory: a new operator is also used to allocate a block(an array) of memory of type *data type*.

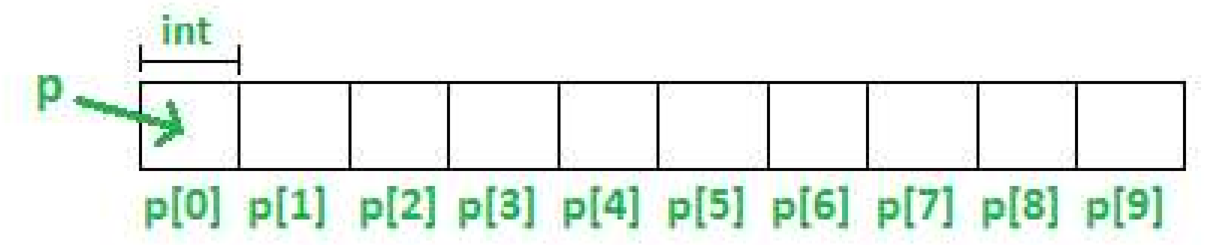
```
pointer-variable = new data-type[size];
```

where size(a variable) specifies the number of elements in an array.

Example:

```
int *p = new int[10]
```

Dynamically allocates memory for 10 integers continuously of type int and returns a pointer to the first element of the sequence, which is assigned top(a pointer). p[0] refers to the first element, p[1] refers to the second element, and so on.



Normal Array Declaration vs Using new

There is a difference between declaring a normal array and allocating a block of memory using new. The most important difference is, that normal arrays are deallocated by the compiler (If the array is local, then deallocated when the function returns or completes). However, dynamically allocated arrays always remain there until either they are deallocated by the programmer or the program terminates.

What if enough memory is not available during runtime?

failure by throwing an exception of type std::bad_alloc, unless “nothrow” is used with the new operator, in which case it returns a NULL pointer (scroll to section “Exception handling of new operator” in this article). Therefore, it may be a good idea to check for the pointer variable produced by the new before using its program.

```
int *p = new(nothrow) int;
if (!p)
{
   cout << "Memory allocation failed\n";
}
```

delete operator

Since it is the programmer’s responsibility to deallocate dynamically allocated memory, programmers are provided delete operator in C++ language.

Syntax:

```
// Release memory pointed by pointer-variable
delete pointer-variable;
```

Here, the pointer variable is the pointer that points to the data object created by ***new***.

Examples:

```
delete p;
delete q;
```

To free the dynamically allocated array pointed by pointer variable, use the following form of *delete*:

```
// Release block of memory
// pointed by pointer-variable
delete[] pointer-variable;

Example:

   // It will free the entire array
```

```
    // pointed by p.
    delete[] p;
```

CPP

```
// C++ program to illustrate dynamic allocation
// and deallocation of memory using new and delete
#include <iostream>
using namespace std;

int main ()
{
    // Pointer initialization to null
    int* p = NULL;

    // Request memory for the variable
    // using new operator
    p = new(nothrow) int;
    if (!p)
        cout << "allocation of memory failed\n";
    else
    {
        // Store value at allocated address
        *p = 29;
        cout << "Value of p: " << *p << endl;
    }

    // Request block of memory
    // using new operator
    float *r = new float(75.25);

    cout << "Value of r: " << *r << endl;

    // Request block of memory of size n
    int n = 5;
    int *q = new(nothrow) int[n];

    if (!q)
        cout << "allocation of memory failed\n";
    else
    {
        for (int i = 0; i < n; i++)
            q[i] = i+1;

        cout << "Value store in block of memory: ";
    }

    // freed the allocated memory
    delete p;
    delete r;

    // freed the block of allocated memory
    delete[] q;

    return 0;
}
```

Output

```
Value of p: 29
Value of r: 75.25
Value store in block of memory: 1 2 3 4 5
```

Time Complexity: O(n), where n is the given memory size.

Related Articles:

- Quiz on new and delete
- delete vs free

This article is contributed by **Akash Gupta**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Like 423

Start Your Coding Journey Now!

Previous Next

Related Articles

1. new vs operator new in C++
2. Difference between Static and Dynamic Memory Allocation in C
3. If memory allocation using new is failed in C++ then how it should be handled?
4. Overloading New and Delete operator in c++
5. new vs malloc() and free() vs delete in C++
6. What is Dynamic Memory Allocation?
7. What are the Operators that Can be and Cannot be Overloaded in C++?
8. Increment and Decrement Operators in Python
9. Conditionally assign a value without using conditional and arithmetic operators
10. Case conversion (Lower to Upper and Vice Versa) of a string using BitWise operators in C/C++

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Improved By : Abhi1909, BabisSarantoglou, jeetundaviya20017, dvsingla28, sackshamsharmaintern, kanheremahesh1729, harendrakumar123

Article Tags : cpp-pointer, C++, School Programming

Practice Tags : CPP

Improve Article Report Issue

Courses

News

Top News

Technology

Work & Career

Business

Finance

Lifestyle

Knowledge

Languages

Python

Java

CPP

Golang

C#

SQL

Kotlin

Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Bootstrap

ReactJS

NodeJS

Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship

tart Your Coding Journey Now!