**What is Memory Allocation NEW keyword in C++**

**Motivation comes from:**

**CVisionDlg& CVisionSingleton::getInstance()**

**{**

**if(m\_pVisionDlg == NULL)**

**{**

**CSingleLock lock(&m\_mutex, true);**

**if(m\_pVisionDlg == NULL) // Ensure still NULL upon lock release**

**m\_pVisionDlg.reset(new CVisionDlg());**

**// unlock the resource for others.**

**lock.Unlock();**

**}**

**return \*m\_pVisionDlg;**

**Or another example:**

**Header:**

static Logger \*m\_pInstance ;

Logger& Logger::Instance() {

//Double-check locking pattern

if (m\_pInstance == nullptr) {

m\_Mtx.lock();

if (m\_pInstance == nullptr) {

m\_pInstance = new Logger{};

}

m\_Mtx.unlock();

}

return \*m\_pInstance ;

**//===============================================================================**

The Complete C\_Reference\_4e.pdf

The **new** operator allocates memory and returns a pointer to the start of it. The

**delete** operator frees memory previously allocated using **new**. The general forms of

**new** and **delete** are shown here:

*p\_var* = new *type*;

delete *p\_var*;

Here, *p\_var* is a pointer variable that receives a pointer to memory that is large enough

to hold an item of type *type*.

*p\_var* = new *var\_type (initializer);*

int main()

{

int \*p;

try {

p = new int (87); // initialize to 87

}

catch (bad\_alloc xa) {

cout << "Allocation Failure\n";

return 1;

}

cout << "At " << p << " ";

cout << "is the value " << \*p << "\n";

delete p;

return 0;

}

From <https://www.geeksforgeeks.org/new-vs-operator-new-in-cpp/#:~:text=new%20keyword,memory%20to%20the%20pointer%20variable>.

The new operator is an **operator** which denotes a request for memory allocation on the Heap. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable. When you create an object of class using new keyword(normal new).

* The memory for the object is allocated using **operator new** from heap.
* The constructor of the class is invoked to properly initialize this memory.

A comple Guide to Prog in C++

Pg 454 Chap 21 – **Dynamic Memory Allocation**

**// Dynamic objects of type long and double**

// ------------------------------------------------------

long \*ptr\_long;

ptr\_long = new long; // No initialization

// of the long object.

\*ptr\_long = 1234567; // Assign a value

double \*ptr\_double;

double z = 1.9;

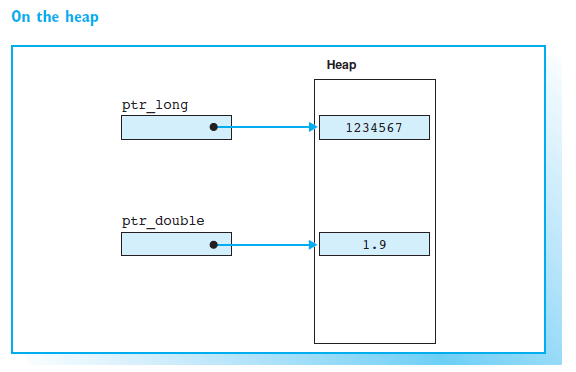
ptr\_double = new double(z); // With initialization

++(\*ptr\_double); // Increment the value

\*ptr\_double += \*ptr\_long; // ok to add long value

ptr\_long = new double(2.7); // Error: ptr\_long not

// pointing to double!



//=========================================================================