**Operators delete and delete[].**

In most cases, memory allocated dynamically is only needed during specific periods of time within a program; once it is no longer needed, it can be freed so that the memory becomes available again for other requests of dynamic memory. This is the purpose of operator *delete*, whose syntax is:

|  |  |  |
| --- | --- | --- |
| 1 2 | delete pointer;  delete[] pointer; |  |

The first statement releases the memory of a single element allocated using *new*, and the second one releases the memory allocated for arrays of elements using *new* and a size in brackets ([]).

The value passed as argument to delete shall be either a pointer to a memory block previously allocated with *new*, or a *null pointer* (in the case of a *null pointer*, delete produces no effect).

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 | // rememb-o-matic  #include <iostream>  #include <new>  using namespace std;  int main ()  {  int i,n;  int \* p;  cout << "How many numbers would you like to type? ";  cin >> i;  p= new (nothrow) int[i];  if (p == nullptr)  cout << "Error: memory could not be allocated";  else  {  for (n=0; n<i; n++)  {  cout << "Enter number: ";  cin >> p[n];  }  cout << "You have entered: ";  for (n=0; n<i; n++)  cout << p[n] << ", ";  delete[] p;  }  return 0;  } | How many numbers would you like to type? 5  Enter number : 75  Enter number : 436  Enter number : 1067  Enter number : 8  Enter number : 32  You have entered: 75, 436, 1067, 8, 32, | [Edit & Run](https://www32.cplusplus.com/doc/tutorial/dynamic/) |

Notice how the value within brackets in the *new* statement is a variable value entered by the user (i), not a constant expression:

|  |  |  |
| --- | --- | --- |
|  | p= new (nothrow) int[i]; |  |

There always exists the possibility that the user introduces a value for *i* so big that the system cannot allocate enough memory for it. For example, when I tried to give a value of 1 billion to the "How many numbers" question, my system could not allocate that much memory for the program, and I got the text message we prepared for this case (Error: memory could not be allocated).

It is considered good practice for programs to always be able to handle failures to allocate memory, either by checking the pointer value (*if nothrow*) or by catching the proper exception.