**sizeof.**

This operator accepts one parameter, which can be either a type or a variable, and returns the size in bytes of that type or object:

|  |  |  |
| --- | --- | --- |
|  | x = sizeof (char); |  |

Here, x is assigned the value 1, because *char* is a type with a size of one byte.

The value returned by *sizeof* is a compile-time constant, so it is always determined before program execution.

When *sizeof()* is used with the expression, it returns size of the expression. Let see example:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | #include <iostream>  using namespace std;  int main()  {  cout << sizeof(char) << "\n";  cout << sizeof(int) << "\n";  cout << sizeof(float) << "\n";  cout << sizeof(double) << "\n";  return 0;  } | 1  4  4  8 | [Edit & Run](https://cplusplus.com/doc/tutorial/operators/) |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 | #include <iostream>  using namespace std;  int main()  {  int a = 0;  double d = 10.21;  cout << sizeof(a + d);  return 0;  } | 8 | [Edit & Run](https://cplusplus.com/doc/tutorial/operators/) |

Another example:

As we know from first case size of *int* and *double* is 4 and 8 respectively, *a* is *int* variable while *d* is a *double* variable. The final result will be a *double*, because one of the operands contains floating point number which is declared and initialized by a double and any operation (adding, subtracting, multiplying, dividing etc.) will lead to the resulting number which is in floating point as well. Thence the output of our program is 8 bytes.

Another use of *sizeof*:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 | #include <iostream>  using namespace std;  int main()  {  int y;  int x = 11;  y = sizeof(x++); //value of x doesn't change  cout << y << " " << x;// prints 4 11  } | 4 11 | [Edit & Run](https://cplusplus.com/doc/tutorial/operators/) |

The purpose of using *sizeof*:

|  |  |
| --- | --- |
| int\* ptr = (int\*)malloc(10 \* sizeof(int)); |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9  10 | #include <iostream>  using namespace std;  int main()  {  int arr[] = { 1, 2, 3, 4, 7, 98,  0, 12, 35, 99, 14 };  cout << "Number of elements: "  << (sizeof(arr) / sizeof(arr[0]));  return 0;  } | Number of elements: 11 | [Edit & Run](https://cplusplus.com/doc/tutorial/operators/) |

More about arrays and pointers later.

As said before in previous chapters, *signed short int* can be abbreviated as *signed short*, *short int*, or simply *short*; they all identify the same fundamental type.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10  11  12  13 | #include <iostream>  using namespace std;  int main()  {  short x = 1;  short int h = 8;  signed short k = 20;  signed short int b = 80;  cout << sizeof(x) << "\t" << x << endl;  cout << sizeof(h) << "\t" << h << endl;  cout << sizeof(k) << "\t" << k << endl;  cout << sizeof(b) << "\t" << b << endl;  } | 2 1  2 8  2 20  2 80 | [Edit & Run](https://cplusplus.com/doc/tutorial/operators/) |