2.17 Numeric data types

int and double are the most common numeric data types. However, several other numeric types exist. The following table summarizes available integer numeric data types.

The size of integer numeric data types can vary between compilers, for reasons beyond our scope. The following table lists the sizes for numeric integer data types used in this material along with the minimum size for those data types defined by the language standard.

Table 2.17.1: Integer numeric data types.

Declaration	Size	Supported number range	Standard-defined minimum size
char myVar;	8 bits	-128 to 127	8 bits
short myVar;	16 bits	-32,768 to 32,767	16 bits
long myVar;	32 bits	-2,147,483,648 to 2,147,483,647	32 bits
long long myVar;	64 bits	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	64 bits
int myVar;	32 bits	-2,147,483,648 to 2,147,483,647	*16 bits*

Feedback?

int is the most commonly used integer type. int

long long is used for integers expected to exceed about 2 billion. That is not a typo; the word appears twice.

In case the reader is wondering, the language does not have a simple way to print numbers with commas. So if x is 8000000, printing 8,000,000 is not trivial.

A common error made by a program's user is to input the wrong type, such as inputting a string like twenty (rather than 20) when the input statement was `cin >> myInt;` where myInt is an int, which can cause strange program behavior.

short is rarely used. One situation is to save memory when storing many (e.g., tens of thousands) of smaller numbers, which might occur for arrays (another section). Another situation is in *embedded* computing systems having a tiny processor with little memory, as in a hearing aid or TV remote control. Similarly, char, while technically a number, is rarely used to directly store a number, except as noted for short.

PARTICIPATION ACTIVITY 2.17.1: Integer types.

Indicate whether each is a good variable declaration for the stated purpose, assuming int is usually used for integers, and long long is only used when absolutely necessary.

1) The number of days of school per year:
`int numDaysSchoolYear;`

[✓ True] False

Correct

Although the max is 366, int is typically used rather than short.

2) The number of days in a human's lifetime.
`int numDaysLife;`

[✓ True] False

Correct

The number might be as big as about 100 yrs * 365 days/year or 36,500. int can hold up to about 2 billion, so is large enough.

3) The number of cells in the average human body.
`int numCells;`

True [✓ False]

Correct

The human body contains approximately 37.2 trillion cells, exceeding the roughly 2 billion limit of an int. A long long is needed.

4) The number of human heartbeats in one year, assuming 100 beats/minute.
`long long numHeartBeats;`

True [✓ False]

Correct

100 beats/min * 60 min/hr * 24 hr/day * 365 days/yr yields 52,560,000 beats/yr. 52 million is much less than an int's 2 billion max. A long long is not necessary.

Feedback?

The following table summarizes available floating-point numeric types.

Table 2.17.2: Floating-point numeric data types.

Declaration	Size	Supported number range
float x;	32 bits	$-3.4\text{x}10^{38}$ to $3.4*10^{38}$
double x;	64 bits	$-1.7\text{x}10^{308}$ to $1.7*10^{308}$

Feedback?

The compiler uses one bit for sign, some bits for the mantissa, and some for the exponent. Details are beyond our scope. The language (unfortunately) does not actually define the number of bits for float and double types, but the above sizes are very common.

float is typically only used in memory-saving situations, as discussed above for short.

Due to the fixed sizes of the internal representations, the mantissa (e.g, the 6.02 in 6.02e23) is limited to about 7 significant digits for float and about 16 significant digits for double. So for a variable declared as double pi, the assignment pi = 3.14159265 is OK, but pi = 3.14159265358979323846 will be truncated.

A variable cannot store a value larger than the maximum supported by the variable's data type. An ***overflow*** occurs when the value being assigned to a variable is greater than the maximum value the variable can store. Overflow with floating-point results in infinity. Overflow with integer is discussed elsewhere.

PARTICIPATION ACTIVITY 2.17.2: Representation of 32-bit floating-point values.

Enter a decimal value:

0.124134

Convert

0.124134 in a 32-bit floating-point representation (IEEE):

Sign	Exponent	Mantissa
0	0 1 1 1 1 0 1 1	1. 1 1 1 1 1 1 0 0 0 1 1 1 0
+1	$2^{(123 - 127)} = 2^{-4}$	1.98614394664

Representation's value: 0.12413399666547775

Feedback?

On some processors, especially low-cost processors intended for "embedded" computing, like systems in an automobile or medical device, floating-point calculations may run slower than integer calculations, such as 100 times slower. Floating-point types are typically only used when really necessary. On more powerful processors like those in desktops, servers, smartphones, etc., special floating-point hardware nearly or entirely eliminates the speed difference.

Floating-point numbers are sometimes used when an integer exceeds the range of the largest integer type.

PARTICIPATION ACTIVITY 2.17.3: Floating-point numeric types. ✓

1) float is the most commonly-used floating-point type.

True [✓ False]

✓ **Correct**

Today, double is. Decades ago, float was, but then larger computer memories allowed programmers to use a floating-point type that was "double" the size of float, hence the name.

2) int and double types are limited to about 16 digits.

True [✓ False]

✓ **Correct**

A double's mantissa is limited to 16 digits (due to the number of bits that represent the mantissa), but an int's range is about +/- 2 billion which means 10 digits.

Feedback?

(*int) Unfortunately, int's size is the processor's "natural" size, and not necessarily 32 bits. Fortunately, nearly every compiler allocates at least 32 bits for int.