# 2.12 Type conversions

## Type conversions

A calculation sometimes must mix integer and floating-point numbers. For example, given that about 50.4% of human births are males, then `0.504 * numBirths` calculates the number of expected males in numBirths births. If numBirths is an int variable (int because the number of births is countable), then the expression combines a floating-point and integer.

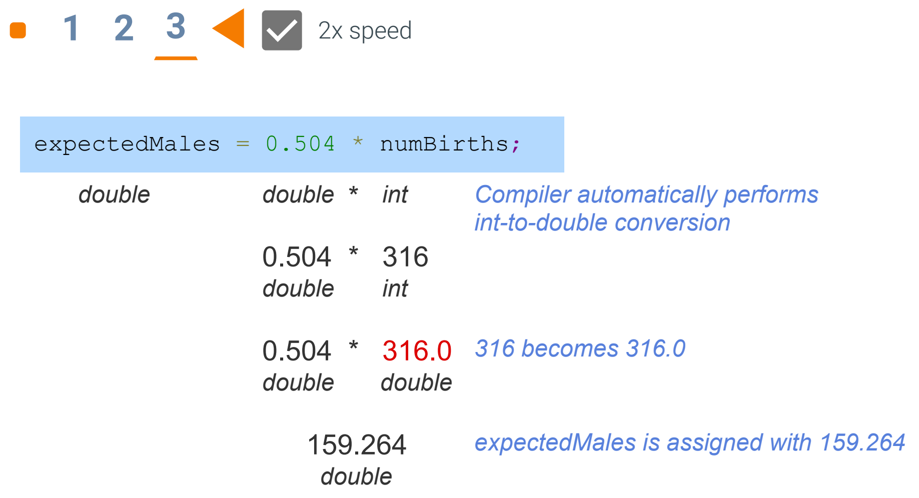
A ***type conversion*** is a conversion of one data type to another, such as an int to a double. The compiler automatically performs several common conversions between int and double types, such automatic conversion known as ***implicit conversion***.

- For an arithmetic operator like + or *, if either operand is a double, the other is automatically converted to double, and then a floating-point operation is performed.
- For assignments, the right side type is converted to the left side type.

*int-to-double* conversion is straightforward: 25 becomes 25.0.

*double-to-int* conversion just drops the fraction: 4.9 becomes 4.

PARTICIPATION ACTIVITY 2.12.1: Implicit type conversion: int-to-double.



```
expectedMales = 0.504 * numBirths;
```

Then, the program computes 0.504 * 316.0 yielding 159.264. expectedMales is a double variable and is assigned with that result.

Feedback?

PARTICIPATION ACTIVITY 2.12.2: Implicit conversions among double and int.

Type the value of the expression given int numItems = 5. For any floating-point answer, type answer to tenths. Ex: 8.0, 6.5, or 0.1.

1) 3.0 / 1.5

2.0

Check Show answer

**Correct**

2.0

Normal floating-point division.

✓

2) 3.0 / 2

1.5

Check Show answer

**Correct**

1.5

2 is first converted to 2.0, then 3.0 / 2.0 yields 1.5.

✓

3) (numItems + 10) / 2

7

Check Show answer

**Correct**

7

5 + 10 is 15. 15 / 2 yields 7 (remainder of 1 is ignored).

✓

4) (numItems + 10) / 2.0

7.5

Check Show answer

**Correct**

7.5

5 + 10 is 15. That int is converted to double 15.0. Then 15.0 / 2.0 yields 7.5.

✓

Feedback?

**PARTICIPATION ACTIVITY** 2.12.3: Implicit conversions among double and int with variables. ✓

Type the value held in the variable after the assignment statement, given int numItems = 5, and double itemWeight = 0.5. For any floating-point answer, type answer to tenths. Ex: 8.0, 6.5, or 0.1

1) someDoubleVar = itemWeight * numItems; (someDoubleVar is type double).

2.5

Check Show answer

**Correct**

2.5

5 is converted to 5.0. Then 0.5 * 5.0 yields 2.5, which is assigned to someDoubleVar.

✓

2) someIntVar = itemWeight * numItems; (someIntVar is type int).

2

Check    Show answer

**Correct**

2

5 is converted to 5.0. Then 0.5 * 5.0 yields 2.5. The compiler sees "int = double" so converts the double to int by truncating the fraction, yielding 2.

Feedback?

## Assigning doubles with integer literals

*Because of implicit conversion, statements like* `double someDoubleVar = 0;` *or* `someDoubleVar = 5;` *are allowed, but discouraged. Using 0.0 or 5.0 is preferable.*

## Type casting

A programmer sometimes needs to explicitly convert an item's type. Ex: If a program needs a floating-point result from dividing two integers, then at least one of the integers needs to be converted to double so floating-point division is performed. Otherwise, integer division is performed, evaluating to only the quotient and ignoring the remainder. A ***type cast*** explicitly converts a value of one type to another type.

The ***static_cast*** operator (`static_cast<type>(expression)`) converts the expression's value to the indicated type. Ex: If myIntVar is 7, then `static_cast<double>(myIntVar)` converts int 7 to double 7.0.

The program below casts the numerator and denominator each to double so floating-point division is performed (actually, converting only one would have worked).

## Figure 2.12.1: Using type casting to obtain floating-point division.

```
Average kids per family:
3.5
```

```
#include <iostream>
using namespace std;

int main() {
   int kidsInFamily1;       // Should be int, not double
   int kidsInFamily2;       // (know anyone with 2.3 kids?)
   int numFamilies;

   double avgKidsPerFamily; // Expect fraction, so double

   kidsInFamily1 = 3;
   kidsInFamily2 = 4;
   numFamilies = 2;

   avgKidsPerFamily = static_cast<double>(kidsInFamily1 +
kidsInFamily2)
                      / static_cast<double>(numFamilies);

   cout << "Average kids per family: " << avgKidsPerFamily << endl;

   return 0;
}
```

Feedback?

PARTICIPATION ACTIVITY

## 2.12.4: Type casting.

Determine the resulting type for each expression. Assume numSales1, numSales2, and totalSales are int variables.

1) (numSales1 + numSales2) / 2

● int

○ double

**Correct**

numSales1 + numSales2 yields an int. Dividing the int sum by the integer literal 2 yields an int value.

2) static_cast<double>(numSales1 + numSales2) / 2

○ int

● double

**Correct**

numSales1 + numSales2 yields an int. The int sum is then cast to a double value. Dividing a double by an int causes the int to be implicitly converted to a double, resulting in a double value.

3) (numSales1 + numSales2) / totalSales

● int

○ double

**Correct**

numSales1 + numSales2 yields an int. Dividing the int sum by totalSales (an integer variable) yields an int value.

4) (numSales1 +

**Correct**

numSales2) / static_cast<double> (totalSales)

○ int

● double

numSales1 + numSales2 yields an int. Then, totalSales is cast to a double. Dividing the int sum by a double causes the int sum to be implicitly converted to a double, resulting in a double value.

Feedback?

## Common errors

A common error is to accidentally perform integer division when floating-point division was intended. The program below undesirably performs integer division rather than floating-point division.

Figure 2.12.2: Common error: Forgetting cast results in integer division.

```
#include <iostream>
using namespace std;

int main() {
   int kidsInFamily1;       // Should be int, not double
   int kidsInFamily2;       // (know anyone with 2.3 kids?)
   int numFamilies;

   double avgKidsPerFamily; // Expect fraction, so double

   kidsInFamily1 = 3;
   kidsInFamily2 = 4;
   numFamilies = 2;

   avgKidsPerFamily = (kidsInFamily1 + kidsInFamily2) /
numFamilies;

   // Should be 3.5, but is 3 instead
   cout << "Average kids per family: " << avgKidsPerFamily << endl;

   return 0;
}
```

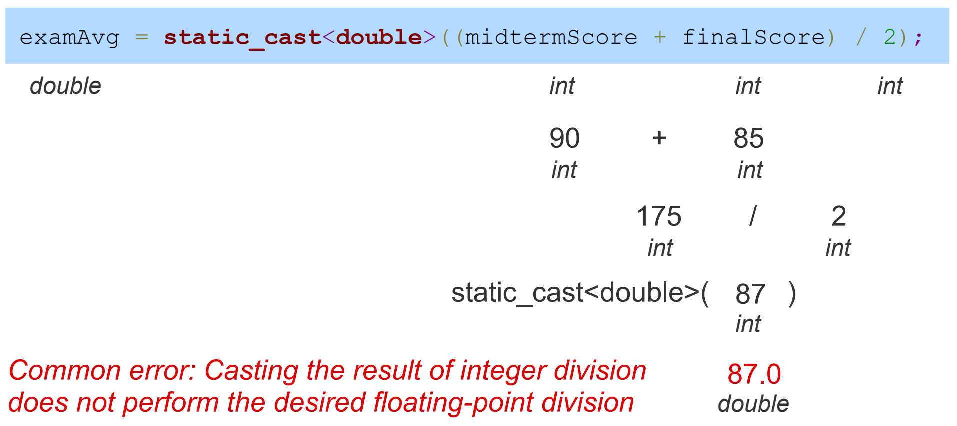
```
Average kids per family: 3
```

Feedback?

Another common error is to cast the entire result of integer division, rather than the operands, thus not obtaining the desired floating-point division.

PARTICIPATION ACTIVITY | 2.12.5: Common error: Casting final result instead of operands. ✓

■ 1 2 3 4 ◀ ☑ 2x speed



```
examAvg = static_cast<double>((midtermScore + finalScore) / 2);
```

double
int
int
int
90 + 85
int int
175 / 2
int int
static_cast<double>( 87 )
int
Common error: Casting the result of integer division does not perform the desired floating-point division
87.0
double

The type cast converts 87 to 87.0. Casting the result of integer division does not perform the desired floating-point division.

Feedback?

**PARTICIPATION ACTIVITY** 2.12.6: Type casting. ✓

1) Which yields 2.5?

○ static_cast<int>(10) / static_cast<int>(4)

● static_cast<double>(10) / static_cast<double>(4)

○ static_cast<double>(10 / 4)

**Correct**

The casts yield 10.0 / 4.0, which is 2.5.

✓

2) Which does NOT yield 3.75?

○ static_cast<double>(15) / static_cast<double>(4)

○ static_cast<double>(15) / 4

○ 15 / static_cast<double>

**Correct**

This common error first does integer division of 15 / 4 which is 3, then converts to 3.0.

✓

(4)

● static_cast<double> (15 / 4)

3) Given aCount, bCount, and cCount are integer variables, which variable must be cast to a double for the expression `(aCount * bCount) / cCount` to evaluate to a double value?

○ None

○ All variables

● Only one variable

**Correct**

Casting any one of the three variables results in floating-point division that yields a double value.

Feedback?

**CHALLENGE ACTIVITY** 2.12.1: Type casting: Computing average kids per family

Compute the average kids per family. Note that the integers should be type cast to doubles.

```
#include <iostream>
using namespace std;

int main() {
   int numKidsA;
   int numKidsB;
   int numKidsC;
   int numFamilies;
   double avgKids;

   cin >> numKidsA;
   cin >> numKidsB;
   cin >> numKidsC;
   cin >> numFamilies;

   /* Your solution goes here  */
   avgKids = (numKidsA + numKidsB + numKidsC)/static_cast<double>(numFamilies);

   cout << avgKids << endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with 1, 4, 5 and numFamilies = 3

Your value
```
3.3333333333333335
```

✓ Testing with 2, 2, 4 and numFamilies = 3

Your value
```
2.6666666666666665
```

✓ Testing with 2, 5, 6 and numFamilies = 4

Your value
```
3.25
```

**Feedback?**