Arithmetic in C++

CS1A

- Arithmetic Operators
- Integer Expressions
- Floating Point Expressions
- Mixed-Mode Arithmetic
- Type Coercion
- Type Casting

Quick Review

```
int main()
{
    const float DENOM = 2.0;

    int     num1;
    int     num2;
    double average;

    cout << "Enter first value: ";
    cin  >> num1;

    cout << "Enter second value: ";
    cin  >> num2;

    average = (num1 + num2) / DENOM;

    cout << "\nThe average is " << average;

    return 0;
}
```

Arithmetic Operators in C++

Symbol	Function
*	Multiplication
/	Division
%	Modulus (remainder from integer division)
+	Addition
-	Subtraction

Order of Precedence
()
* / %
+ -

Expressions are evaluated from left to right according to the order of precedence.

Integer Expressions

When all of the operands in an expression are integers
→ i.e. the expression is all integers and the result is placed in an integer or output

How would these expressions be evaluated?

Examples:
4 + 8 / 2 =
(4 + 8) / 2 =
7 / 3 =
7 % 3 =
12 / 3 * 5 =
10 % 3 – 6 / 2 =
10 % 3.0 =

Order of Precedence
()
* / %
+ -

More Examples
int avg, inum1, inum2;

inum1 = 10;
inum2 = 15;

avg = (inum1 + inum2)/2; ← What is the value of avg?

Floating Point Expressions

- When all the operands are floats and the result is placed in a float or output

Examples:
5.0 * 2.0 / 4.0 * 3.0 =
5.0 * 2.0 / (4.0 * 2.0) =
5.0 + 2.0 / (4.0 * 2.0) =

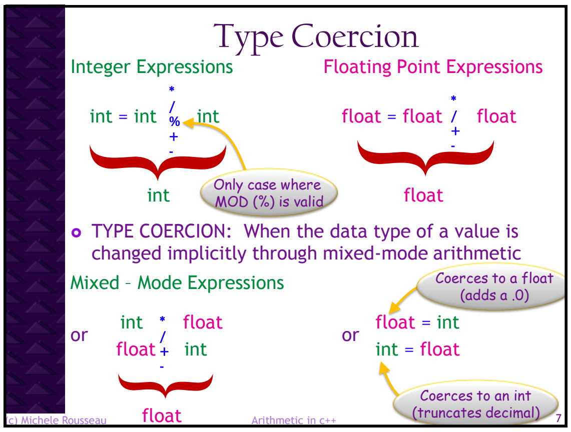
More Examples
float fnum1, fnum2;
double avg;

fnum1 = 10.0;
fnum2 = 15.0;

avg= (fnum1 + fnum2)/2.0; ← What is the value of avg?
= (10.0 + 15.0) / 2.0
= 25.0 / 2.0 = 12.5

Mixed Mode Arithmetic

- Two types of data types to represent numeric values
  - int & float
  - They store data differently
  - Allocate memory differently
    - i.e. int 6 is stored differently than float 6.0
- Mixed mode arithmetic
  - → when we combine different data types
  - e.g. float & int



Type Coercion

Integer Expressions

$\underbrace{\text{int} = \text{int} \; {}^{*}_{/\,\%\,+\,-} \; \text{int}}_{\text{int}}$

Only case where MOD (%) is valid

Floating Point Expressions

$\underbrace{\text{float} = \text{float} \; {}^{*}_{/\,+\,-} \; \text{float}}_{\text{float}}$

- TYPE COERCION: When the data type of a value is changed implicitly through mixed-mode arithmetic

Mixed - Mode Expressions

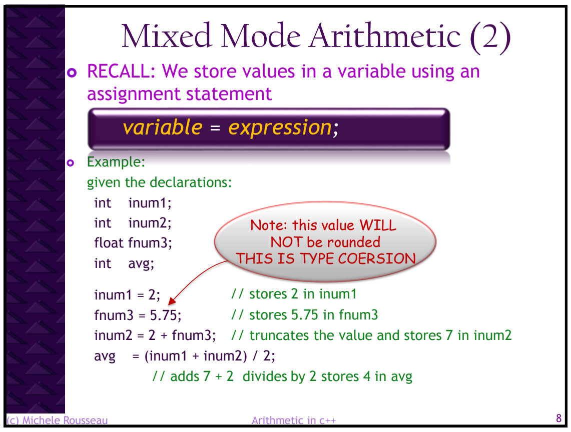
$\underbrace{\text{int} \; {}^{*}_{/\,+\,-} \; \text{float} \quad \text{or} \quad \text{float} \; \text{int}}_{\text{float}}$

Coerces to a float (adds a .0)

or float = int

int = float

Coerces to an int (truncates decimal)



Mixed Mode Arithmetic (2)

- RECALL: We store values in a variable using an assignment statement

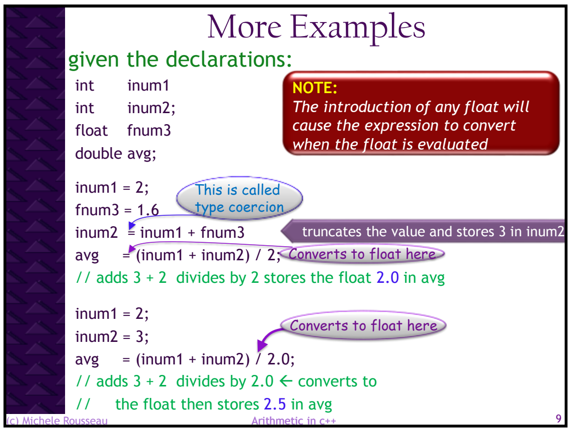
variable = expression;

- Example:

given the declarations:

```
int   inum1;
int   inum2;
float fnum3;
int   avg;

inum1 = 2;              // stores 2 in inum1
fnum3 = 5.75;           // stores 5.75 in fnum3
inum2 = 2 + fnum3;      // truncates the value and stores 7 in inum2
avg   = (inum1 + inum2) / 2;
            // adds 7 + 2  divides by 2 stores 4 in avg
```

Note: this value WILL NOT be rounded THIS IS TYPE COERSION



More Examples

given the declarations:

```
int    inum1
int    inum2;
float  fnum3
double avg;

inum1 = 2;
fnum3 = 1.6
inum2 = inum1 + fnum3
avg   = (inum1 + inum2) / 2;
// adds 3 + 2  divides by 2 stores the float 2.0 in avg

inum1 = 2;
inum2 = 3;
avg   = (inum1 + inum2) / 2.0;
// adds 3 + 2  divides by 2.0 ← converts to
//    the float then stores 2.5 in avg
```

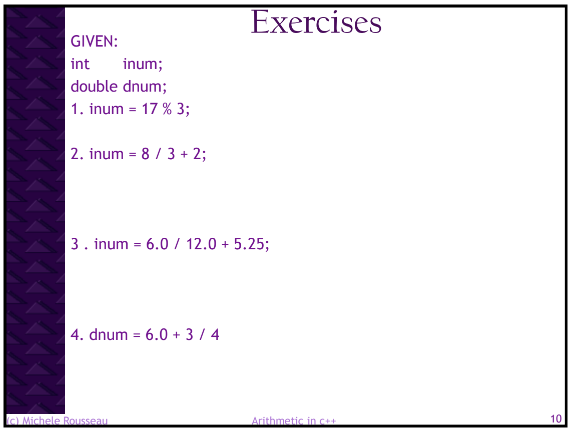
NOTE: *The introduction of any float will cause the expression to convert when the float is evaluated*

This is called type coercion

truncates the value and stores 3 in inum2

Converts to float here

Converts to float here

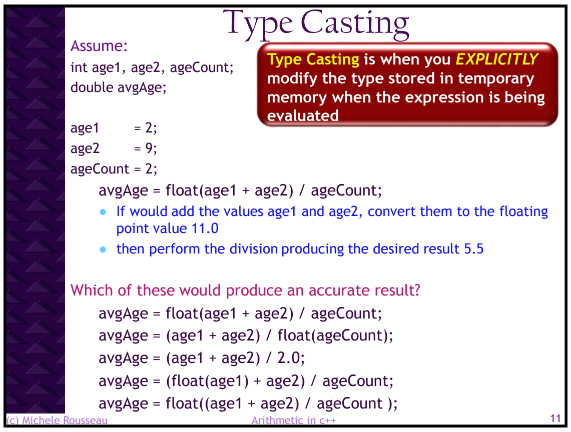


Exercises

GIVEN:

```
int     inum;
double dnum;
```

1. inum = 17 % 3;
2. inum = 8 / 3 + 2;
3. inum = 6.0 / 12.0 + 5.25;
4. dnum = 6.0 + 3 / 4



Type Casting

Assume:

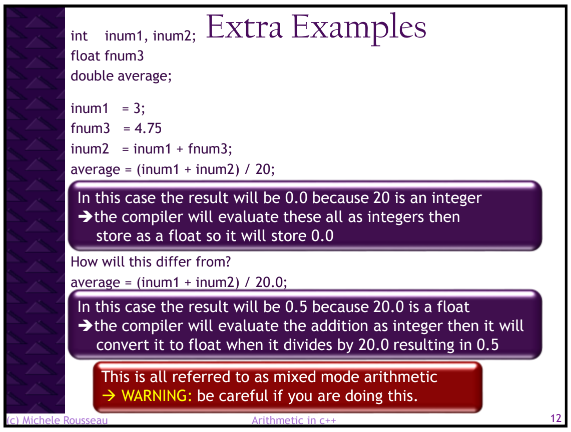
```
int age1, age2, ageCount;
double avgAge;

age1     = 2;
age2     = 9;
ageCount = 2;
    avgAge = float(age1 + age2) / ageCount;
```

Type Casting is when you *EXPLICITLY* modify the type stored in temporary memory when the expression is being evaluated

- If would add the values age1 and age2, convert them to the floating point value 11.0
- then perform the division producing the desired result 5.5

Which of these would produce an accurate result?

```
avgAge = float(age1 + age2) / ageCount;
avgAge = (age1 + age2) / float(ageCount);
avgAge = (age1 + age2) / 2.0;
avgAge = (float(age1) + age2) / ageCount;
avgAge = float((age1 + age2) / ageCount );
```



Extra Examples

```
int   inum1, inum2;
float fnum3
double average;

inum1  = 3;
fnum3  = 4.75
inum2  = inum1 + fnum3;
average = (inum1 + inum2) / 20;
```

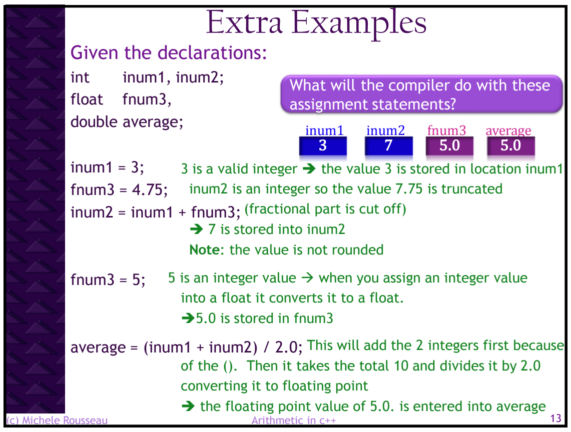
In this case the result will be 0.0 because 20 is an integer
➔the compiler will evaluate these all as integers then store as a float so it will store 0.0

How will this differ from?

average = (inum1 + inum2) / 20.0;

In this case the result will be 0.5 because 20.0 is a float
➔the compiler will evaluate the addition as integer then it will convert it to float when it divides by 20.0 resulting in 0.5

This is all referred to as mixed mode arithmetic
→ WARNING: be careful if you are doing this.



Extra Examples

Given the declarations:

```
int      inum1, inum2;
float    fnum3,
double average;
```

What will the compiler do with these assignment statements?

inum1	inum2	fnum3	average
3	7	5.0	5.0

inum1 = 3; 3 is a valid integer ➔ the value 3 is stored in location inum1

fnum3 = 4.75; inum2 is an integer so the value 7.75 is truncated

inum2 = inum1 + fnum3; (fractional part is cut off)
➔ 7 is stored into inum2
Note: the value is not rounded

fnum3 = 5; 5 is an integer value → when you assign an integer value into a float it converts it to a float.
➔5.0 is stored in fnum3

average = (inum1 + inum2) / 2.0; This will add the 2 integers first because of the (). Then it takes the total 10 and divides it by 2.0 converting it to floating point
➔ the floating point value of 5.0. is entered into average

(c) Michele Rousseau Arithmetic in c++ 13