Type Casting

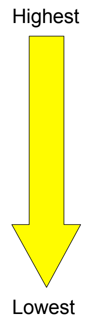
Section 3.3-3.5

Implicit Type Conversion (3.3)

- Mixing the data types of operands during mathematical operations
  - What happens when we save a **`double`** as an **`int`**?
  - What happens when an **`int`** is multiplied by a **`float`**?

Data Type Ranks

`long double`
`double`
`float`
`unsigned long`
`long`
`unsigned int`
`int`



Rules for Type Conversion

- When a value is converted to a higher data type, it is being *promoted*
- When a value is converted to a lower data type, it is being *demoted*
  - Rule 1: **`char`**, **`short`**, and **`unsigned short`** are automatically promoted to **`int`**
  - Rule 2: When an operator works with values of different types, the lower ranking value is promoted to the higher ranking
  - Rule 3: When the value of an expression is assigned to a variable, it is converted to the data type of that variable

Q.2 Practice

- Assume the following variable definitions

```
int a = 5, b = 12;
double x = 3.4;
```

- What are the values of the following expressions:
  a. `b / x`
  b. `x * a`

Explicit Type Conversion (3.4)

- A type cast expression let's you manually change the data type of a value
- The syntax for type casting is
  - **`static_cast<DataType>(Value)`**
  - Value is a variable or literal value
  - DataType is the data type that you are converting Value into

7.3 Example of Type Casting

```
double number = 3.7;
int val;
val = static_cast<int>(number);
```

- What is saved into val?

Uses of Type Casting

- Preventing integer division

```
int books = 30, months = 7;
double booksPerMonth;
booksPerMonth = static_cast<double>(books) / months;
```

  - What about this statement?

```
booksPerMonth = static_cast<double>(books / months);
```

- Displaying a char from its ASCII value

```
int number = 65;
cout << static_cast<char>(number);
```

Q.4 Practice

- What is the value of each of the variables while this expression is being executed?

```
int total;
double gradeCounter, average;
total = 30;
gradeCounter = 4;
average = static_cast<double>(total) / gradeCounter;
```

Overflow and Underflow (3.5)

- What happens when a variable is assigned a value that is too large or too small in range for that variable's data type?

```
short testVar = 32767;
cout << testVar << endl;
testVar = testVar + 1;
cout << testVar << endl;
testVar = testVar - 1;
cout << testVar << endl;
```

```
32767
-32768
32767
```

Multiple Assignments (3.7)

- C++ allows statements such as:

```
a = b = c = d = 45;
```

- Why do you think that is?
- What is the associativity of the assignment operator?

Combined Assignments

- The same variable can be used on the left hand side of the assignment and on the right hand side

```
notes = notes / 20;
note = notes % 20;
```

- These are common in programming, so the two operators can be combined as follows:

```
notes /= 20;
note %= 20;
```

Examples of Combined Assignments

Operator	Example Usage	Equivalent To
`+=`	`x += 5;`	`x = x + 5;`
`-=`	`y -= 2;`	`y = y - 2;`
`*=`	`z *= 10;`	`z = z * 10;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`c %= 3;`	`c = c % 3;`

Q.5 Combined Assignments

- Combined assignments can be combined with arithmetic operators

a. `y -= a * 2;`

b. `a /= b + c;`

c. `C %= d - 3;`

- What is the long form of these statements?

Q.6 What is the Output?

```
int unus, duo, tres;

unus = duo = tres = 5;
unus += 4;
duo *= 2;
tres -= 4;
unus /= 3;
duo += tres;
cout << unus << endl;
cout << duo << endl;
cout << tres << endl;
```

Q.1 Practice

- Write a C++ program that allows the user the ability to enter the number of nickels and pennies they have. You are then to print the number of dollars and change that corresponds to. The change should be in the form of nickels and pennies