Data Types

Ever feel like you’re a floating-point peg in an integer hole?

CMPT 130
© Dr. B. Fraser
Topics

1) How can we store numbers like 3.14?
2) How can we convert between numbers like: 3.14 to 3?
Floating Point
Floating Point

• Floating point numbers are used to store values like: 3.1415, -0.03, 0.0000000000001, 6.7 \times 10^{84}

• They are stored using (effectively) scientific notation:
  - 3.1415E0, -3.0E-2, 1.0E-12, 6.7E84

• Types:
  - float ..
    (typically) 7 significant digits, up to 3.4E38
  - double Double precision
    (typically) 16 significant digits, up to 1.7E308
  - long double Often larger than double.

Note no unsigned floating point types.
// Example for floating point numbers
#include <iostream>
using namespace std;

int main()
{
    double distanceSun = 1.49E18; // in km
    double massSun = 1.989E30;    // in kg
    double timeVisible = 12.3;    // in hours

    cout << "The sun is " << distanceSun << " km away.\n";
    cout << "It weighs " << massSun
         << " and we can see it for " << timeVisible << " hours per day.\n";
    return 0;
}

.. Output:

The sun is 1.49e+18 km away.
It weighs 1.989e+30 and we can see it for 12.3 hours per day.
Printing Floating Point Numbers

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

int main() {
    // Read in number of people.
    cout << "Enter # people: ";
    int numPeople = 0;
    cin >> numPeople;

    // Read in the bill amount
    cout << "Enter $";
    double billAmount = 0.0;
    cin >> billAmount;

    // Calculate how much money per person.
    double costPerPerson = billAmount / numPeople;

    // Control floating point display:
    cout << fixed << setprecision(2);

    // Display answer:
    cout << "$" << billAmount << " split " << numPeople
         << " ways is "$ << costPerPerson << " each.\n";
}
```

Output:
Enter # people: 2
Enter $8.10

$8.10 split 2 ways is $4.05 each.

Question: Does cout round?

---

```
fixed and setprecision() do not affect ints; just float/double.
```

---

Force cout to display floating point values to..
Exactly your number?

- What does this print?
  - `cout << fixed << "Exactly? " << 1.2e20;`

  - `fixed` forces it to not be scientific notation.

- What does this print?
  - `cout << fixed << "Exactly? " << 1.2e30;`

- Why?
  - Floating point numbers have..
Comparing floating point

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

int main()
{
    float bankBalance = 0;
    for (int i = 0; i < 10; i++) {
        bankBalance += 0.1;
    }
    cout << "Balance: " << bankBalance << endl;
    if (bankBalance == 1) {
        cout << "Oh! Be 1!\n";
    } else {
        cout << "The dark side.\n";
    }
    cout << fixed << setprecision(30) << "= " << bankBalance << endl;
}
```

Output:

```
More digits
```

- Floating point values are..
1. What data type fits each of these:
   a) 3.1415
   b) 123456
   c) 'a'

2. What is the difference between these?
   cout << 1.1;
   cout << fixed << setprecision(3) << 1.1;

3. What is the 'problem' with this code (besides magic #'s):
   for (float depth = 0; depth != 12; depth += 1.2) {
     cout << “Adding another layer...\n”;
   }
Type conversions
Floating point to integer

- Floating point values hold more information than integer values:
  - How could you “put” 8.254 into an int?

- int num = 8.254;       // num actually holds 8.
- int height = 2.9999;   // height actually holds 2.
- int time = -9.51;      // time actually holds -9.
Rounding and Truncating Positive #'s

float pie = 3.14;
float e = 2.71; // Euler's number
int x, y;

- **Truncate:** Throws away decimal point.
  - x = pie;
    y = e;

- **Round toward -infinity.**
  (Same as truncate for positive numbers)
  - x = floor(pie);
    y = floor(e);

- **.5 and greater rounds up (+ive numbers)**
  - x = round(pie);
    y = round(e);

- **Round toward +infinity.**
  - x = ceil(pie);
    y = ceil(e);
Type ranking

- All types have a rank:
  - The larger the number that it can store, the higher its rank.
- Conversion from a lower rank to a higher rank.
- Conversion from a higher rank to a lower rank.

- Generally you don't lose information in a promotion, but you might in a demotion.

Simplified Type Ranking (Highest on top)
- double
- float
- int
- char

many types omitted
Type Conversions

- Managing types in expressions:
  - All values in C++ have a type.
  - May need to..

  ```
  double distance = 100;  // double <-- int
  ```

- Two Types of conversions:
  - done automatically (above example)
    - Also called type coercion.
  - done by expression in code.
Implicit type conversion rules

1) Operators to higher operand's rank.

- Example:
  int i = 10;
  double d = 1.1;
  cout << (d / i) << (i / d);

- What happens here?
  int i = 1;
  long l = 4;
  float f = 100;
  cout << i / l * f;

Operands to the /'s are double and int.

The int is...

double in both cases

* associates...

  i*l:
  int i promoted to long.

  (i / l) * f:
  (i*l) is of type long, promoted to float.
Implicit type conversion rules

2) Final value of an assignment is..

- May be a promotion or demotion.
  
  ```
  int people = 10, apples = 15;
  float each = apples / people;
  ```

- Floating point to Integer...
  
  ```
  float purchase = 10, tax = 1.12;
  int asInt = purchase * tax;
  float asFloat = purchase * tax;
  ```

- Performs...
  
  ```
  15/10 = 1!
  each = 1.0
  ```

  ```
  10.0 * 1.12 = 11.2.
  asInt = 11
  asFloat = 11.2
  ```

(“1.0” just means it's a float; not how cout will print it.)
Review

1. What is the value/output of each of the following?
   a. int a = 2.987;

   b. float b = 1 / 2;

   c. cout << ('a' + 1);

   d. int d = 1.5 + 1.5;
Explicit type conversion

• Sometimes we want to force the compiler to treat a value as a different type:

    int people = 10, apples = 15;
    float each = apples / people;

    − We would like the answer to be 1.5!
    − Must explicitly cast the value, which forces a promotion or demotion, using static_cast
      each = static_cast<float>(apples) / people;
How much do you want to be paid?

// Calculate your hourly wage from a yearly salary.
#include <iostream>
using namespace std;

int main() {
    // Constants for a working year:
    const int WEEKS_PER_YEAR = 50;
    const int HOURS_PER_WEEK = 40;
    const int HOURS_PER_YEAR = WEEKS_PER_YEAR * HOURS_PER_WEEK;

    // Read in the yearly salary.
    int salary = 0;
    cout << "Enter the yearly salary you would like: $";
    cin >> salary;

    // Calculate the wage and display it.
    float hourlyWage = (salary) / HOURS_PER_YEAR;

    cout << "So, ask for an hourly wage of $" << hourlyWage << "," << endl;
    cout << "you will earn $" << (hourlyWage * HOURS_PER_YEAR) << " per year." << endl;
    return 0;
}
Casting notes

- Casting only...
  ```cpp
  int a = 15, b = 10;
  double x = static_cast<double>(a) / b;  // =
  double y = a / b;  // =
  ```

- Be careful to cast the...
  ```cpp
  double p = static_cast<double>(a) / b;  // =
  double q = a / static_cast<double>(b);  // =
  double r = static_cast<double>(a / b);  // =
  ```

- Other (older) ways to cast
  - Use static_cast in this course, see the text for more.

Comments show the value.
Output to screen, may show differently:
```
cout<<1.0; outputs "1".
```
Math Functions
Exponents

• Use the pow() function from the math library:
  - `#include <cmath>` // In the math library.
  - `result = pow (10, 2);` // $10^2$
  - `result = pow (x+1, y);` // $(x+1)^y$

• pow Function details:
  double pow(double base, double exponent)
Math Functions

- Some math functions in `<cmath>`:

  ```c
  double y = 0;
  a = abs(-10);    // Returns positive value (10)
  y = log10(10.5); // Log base 10.
  y = log(10.5);   // Natural log (ln)
  y = ceil(2.01);  // Ceiling: round up.
  y = sqrt(25.0);  // Square root
  y = sin(1.1);    // sin function. Also tan, cos.
  ```
Summary

• Floating point data type.
  – Formatting floats using fixed and setprecision()

• Truncate vs round vs round up.

• Type Conversion:
  – Implicit type conversions happen automatically.
  – Explicit type conversions by casting
    ...static_cast<double>(10)...

• Math functions like pow(), ceil()