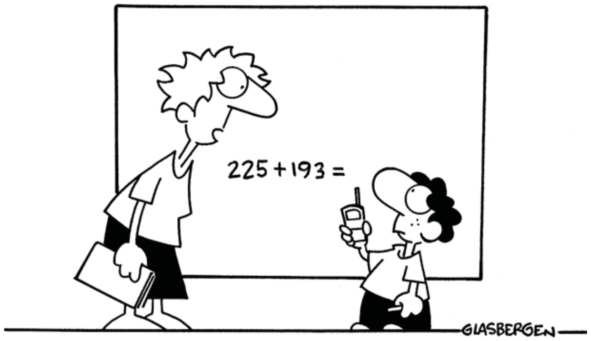
Input and Expressions

Chapter 3
Slides #4

Copyright 2005 by Randy Glasbergen. www.glasbergen.com



"You have to solve this problem by yourself. You can't call tech support."

CMPT 125/128
© Dr. B. Fraser

Topics

1) How can we read data from the keyboard?
2) How can we calculate values?
3) How can we manage the type of a value?
4) How can we round or get random numbers?

Input with cin

Section 3.1

Input

- Almost every computer program needs input.
- Examples:
  - Calculate # pizzas for a party: input # people.
  - Calculate gas mileage: input distance and fuel used.
- Input with cin:...

```
int people = 0;
cin >> people;
```

  - >> is the...
  - cin waits for the user to type in...
  - Places the answer in the given variable.

Prompts

- Prompting the User:
  - cout: Display a prompt to user asking for input.
  - cin: Read keyboard input into a variable.

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

int main() {
    int favNum = 0;

    // Read in user's favourite number:
    cout << "Enter your favourite number: ";
    cin >> favNum;
    cout << "Your favourite number is: " << favNum<<endl;

    return 0;
}
```

```
Enter your favourite number: 42
Your favourite number is: 42
```

Input Example

```
// Ask the user for their personal information.
#include <iostream>
#include <string>          // MUST INCLUDE THIS!!
using namespace std;

int main() {
    string name;          float height;      int speed;

    cout << "What is your name? ";
    cin >> name;
    cout << "What is your height in meters? ";
    cin >> height;
    cout << "What is the airspeed velocity of an unladen swallow? ";
    cin >> speed;

    cout << endl;
    cout << "Hello Sir " << name << ", whose height is " << height << "."<<endl;
    cout << "A swallow's airspeed is NOT " << speed << "!"<<endl;

    return 0;
}
```

Buffered input

- Keyboard data is read into an...
  - cin pulls data out of the buffer as required.

```
// Demonstrate data being left in the buffer.
#include <iostream>
using namespace std;

int main() {
    int age;
    float height;

    cout << "What is your age? ";
    cin >> age;
    cout << "What is your height in meters? ";
    cin >> height;

    cout << endl;
    cout << "Your age is " << age;
    cout << ", and height is " << height<<"."<<endl;

    return 0;
}
```

```
What is your age? 12
What is your height in meters? 2.51

Your age is 12, and height is 2.51.
```

User enters 10.5.
age gets 10, but stops on '.'

so it's read into height.

```
What is your age? 10.5
What is your height in meters?
Your age is 10, and height is 0.5.
```

Chaining

- Chaining:
  - using more than...
    in a statement.
- Examples:
  - cout << "Hello " << "world!" << endl;
  - int width, height, length;
    cin >> width >> height >> length;

Multiple inputs

```
// Demonstrate cin chaining with a rectangle.
#include <iostream>
#include <string>       // NEEDED!
using namespace std;

int main() {
      double length, width;
      string name;

      cout << "Describe a rectangle: "<<endl;
      cout << "Enter: length width name [ENTER]"<<endl;
      cin >> length >> width >> name;

      double area = length * width;
      cout << endl;
      cout << "Box '" << name << "' = " << length << " x ";
      cout << width << ", area is " << area << endl;

      return 0;
}
```

```
Describe a rectangle:
Enter: length width name [ENTER]
2  3.5  Small[ENTER]

Box 'Small' = 2 x 3.5, area is 7
```

cin gives first value to length, then width, then name.

Must be entered in

Review

1. What is the >> operator called?
2. Write a <u>single</u> C++ statement to read in the following two variables:
   int age; float height;
3. True of false: You need to press enter after typing in data being read by a cin statement?

Math Expressions

Section 3.2

(And not like "Wow! Math is great!")

Expressions

- Expression:
  - A statement that...
  - Usually has an operator.
- Examples:

  result = 3;
  result = x * 2;
  result = 1 * x + 2;
- Expressions usable anywhere a value is needed:
  - cout << "Big number " << (1 + 2) << endl;

Order of Operations

- What is the value of result?
  int result = 4 + 10 / 2;
  - Is it 7 or 9? (4 + 10) / 2 or 4 + (10 / 2)
- Each operator is given a precedence:
  - Higher precedence operators are applied first.
  - / is higher than +, so the answer is...
  - Add brackets to force an ordering.
- Associativity:
  - Apply the operators from right-to-left, or left-to-right?
  - +, - are left to right: do the one on the...
  - =, += are right to left: do the one on the...

Operator precedence

- Operators at same

  evaluated based on associativity.
  - * and / from L to R
  - = and += from R to L
- Examples:
  - result = -20 + 9 / 5;
  - result = (-20 + 9) / 5;
  - val = 6 + 5 * 4 / 3 * 2;
  - sum = sum + 10;

Prec. Level	Op.	Operation	Associates
1	[ ]	Array Index	L to R
2	+ -	unary plus unary minus	R to L
3	* / %	mult, div, remainder	L to R
4	+ -	add subtract	L to R
5	<< >>	stream ins. extract.	L to R
6	< <= > >=	comparisons	L to R
7	= += -= *= ...	assignments	R to L

Order can be forced by parentheses.
See text appendix B for full table.

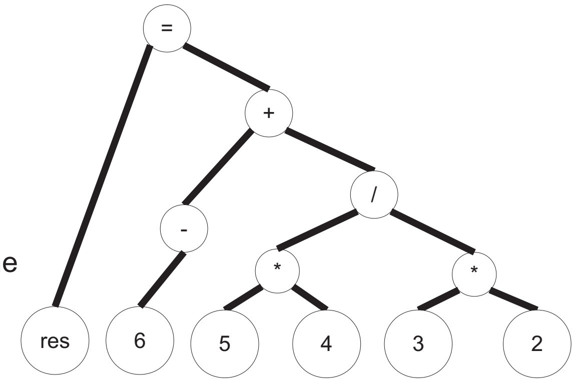
Brackets

- A statement can be correct, but unreadable:
  - result = 1 + 2 / 6 - 1 * 3 / 4 - 3 - -3 * +4;
- Add brackets to make it clear:
  - result = 1 + (2 / 6) - (1 * 3 / 4) - 3 - ((-3) * (+4));

Expression tree

- Represent res = (-6 + 5 * 4 / (3 * 2)) as a tree:

- Operands as leaves.
- Operators as branching nodes.
- Evaluate from the
- Operations lower in the tree have



Review

- Draw an expression tree for the following:

  answer = 5 * x + 6 * (1 – x);

Type conversions

Sections 3.3, 3.4

Type ranking

- All types have a rank:
  - The larger the number that it can store, the higher its rank.
- Type promotion:
  - Conversion from a lower rank to a higher rank.
- Type demotion:
  - Conversion from a higher rank to a lower rank.
- Generally you don't lose information in a promotion, but you might in a demotion.

Type Ranking (Highest on top)
double
float
unsigned long
long
unsigned int
int
unsigned short
short
char

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) char, short, unsigned short promoted to int.

   - Example:

     ```
     char cost = 50;
     short count = 3000;
     int total = cost * count;
     ```

   - This is done to make it 'easier' for the computer to do the computation.
     - The int type is generally setup to be an efficient size for calculations on most machines.

Implicit type conversion rules

2) Operators promote lower rank operand to higher operand's rank.

   - Example:

     ```
     float f = 10.0;
     double d = 1.1;
     cout << (d / f) << (f / d);
     ```

   Operands to / and * are double and float.

   The float is

   double in both cases

   - What happens here?

     ```
     int i = 5;
     long l = 10;
     float f = 100;
     cout << i * l * f;
     ```

   * associates...

   i*l:
   int i promoted to long.

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

Implicit type conversion rules

3) Final value of an assignment is converted to data type of variable.

   - May be a promotion or demotion.

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

   Performs

   15/10 = 1!

   each = 1.0

   - Floating point to Integer...

     ```
     float purchase = 10, tax = 1.12;
     long cost_l = purchase * tax;
     float cost_f = purchase * tax;
     ```

   10.0 * 1.12 = 11.2.

   cost_l = 11
   cost_f = 11.2

Review

1. What is the value 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:
    long WEEKS_PER_YEAR = 50;
    long HOURS_PER_WEEK = 40;
    long HOURS_PER_YEAR = WEEKS_PER_YEAR * HOURS_PER_WEEK;

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

    // Calculate the wage and display it.
    float hourlyWage = static_cast<float>(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;
}
```

```
Enter the yearly salary you would like: $123456
So, ask for an hourly wage of $61.728,
you will earn $123456 per year.
```

hourlyWage.cpp

Casting notes

- Casting only...

```
int a = 15, b = 10;
double x = static_cast<double>(a) / b;     // =
double y = a / b;                          // =
```

- Be careful to cast the...

```
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".

Overflow & Underflow

Section 3.5

Overflow & Underflow

- Each type has a maximum value it can store.
  - Maximum + 1 overflows to the most negative.
  - Minimum – 1 underflows to the most positive.

```
// Work with overflow/underflow
#include <iostream>
using namespace std;

int main() {
    // Demonstrate an overflow/underflow
    short test = 32767;
    cout << "Test starts out at: "<<test<<endl;
    test = test + 1;
    cout << "Adding one gives us: "<<test<<endl;
    test = test - 1;
    cout << "Now subtracting 1:   "<<test<<endl;

    return 0;
}
```

```
Test starts out at: 32767
Adding one gives us: -32768
Now subtracting 1:   32767
```

overflow.cpp

Constants

Section 3.6

Constants

- We have already used literal constants:
  ```
  int x = 10;                  // Numeric constant
  cout << "Hello world!";      // String literal
  ```
- Raw number in code are magic numbers:
  ```
  int h = m / 60;
  long c = s / 72;
  ```
- Use named constants like variables:
  - const int MIN_PER_HOUR = 60;
    int h = s / MIN_PER_HOUR;

const

- const qualifier makes variable...
  ```
  const double TAX_RATE = 0.12;
  const short DAYS_PER_WEEK = 7;
  ```
  - Constants must be given a value when created.
  - Name is upper case by convention.
  - Program cannot modify value of a constant:
    - TAX_RATE = 0.13; // ERROR!
- Advantages:
  - Program becomes more...
  - Can change value in entire program in one spot.
    - Ex: change tax rate that's used in 100 calculations!

Example with const

```
// Work with constants.
#include <iostream>
using namespace std;

int main() {
    const double PI = 3.14159;
    const int DAYS_PER_WEEK = 7;

    double diameter;
    cout << "How big a pizza did you order? ";
    cin >> diameter;

    double radius = (diameter/2);
    double area = PI * radius * radius;
    double pizzaPerDay = area / DAYS_PER_WEEK;
    cout << "You can eat "<<pizzaPerDay
        <<" square inches of pizza per day this week."<<endl;

    return 0;
}
```

```
How big a pizza did you order? 10
You can eat 11.22 square inches of pizza per day this week.
```

pizzaArea.cpp

Guide to Constants

- Which of the following literal constants would be best made into named constants?
  - int numStudents = 0;
  - int next = numStudents + 1;
  - int daNum = numStudents – 72;

Multiple and Combined Assignments

Section 3.7

Assignment Operators

- Combine an operation with assignment:
  - +=, -=, *=, /=, %=
- Examples:
  - a += b;   // means a = a + b;
  - a *= b;   // means a = a * b;
  - a /= 2 + 3;   // means...

Multiple Assignments

- C++ can assign values to multiple variables in one statement:

  -
    ```
    int a = 10, b = 20, c =30;
    a = b = c = 0;        // Set all 3 variables to 0.
    a = b = c = 10*5;     // Set all 3 variables to 50.
    ```
- Basically, (a = b) does two things:
  - sets a to be equal to the value of b; and
  -
    ```
    int x, y = 10;
    cout << (x = y);   // sets x to 10, and outputs "10"
    ```

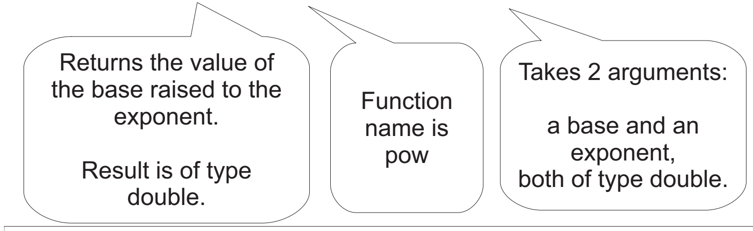
Math Functions

Section 3.11

Exponents

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

  double pow(double base, double exponent)



Returns the value of the base raised to the exponent.

Result is of type double.

Function name is pow

Takes 2 arguments:

a base and an exponent, both of type double.

Area of a pizza

```
// Calculate the area of a pizza
#include <iostream>
#include <cmath>      // NEEDED!
using namespace std;

int main() {
    double diameter;

    cout << "Enter diameter of the pizza: ";
    cin >> diameter;

    // Area is Pi * r^2
    double area = 3.14159 * pow(diameter / 2, 2);
    cout << "Pizza of diameter " << diameter;
    cout << " has area " << area << ".\n";

    return 0;
}
```

```
Enter diameter of the pizza: 18
Pizza of diameter 18 has area 254.469.
```

Math Functions

- Some math functions in <cmath>:

```
int a = 0;
double y = 0;

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

'Random' numbers

- Computers are not Random
  - But we would like random numbers!
- Use rand() to return a pseudorandom integer between 0 and 32767
  - #include <cstdlib>
  - int a = rand();
    int b = rand();
    int c = rand();
- However:
  - Each time the program is run,
    a will have the same value, b will, and c will!

Seed

- The pseudorandom sequence is based on a seed
  - use srand() to seed the sequence once.
    srand(42);
  - Based on a certain seed, the program
- Randomize by the timer
  - Computers have clocks.
  - We can get what seems a very random seed by using the timer:
    #include <ctime>
    srand(time(0));

Dice rolling

```
// Experiment with rand
#include <iostream>
#include <cstdlib>          // NEEDED for rand() and srand()
#include <ctime>            // NEEDED for time()
using namespace std;

int main() {
    // Pick a random seed based on the timer
    srand(time(0));

    // Do a bunch of D20 rolls (1 to 20):
    const int MAX_VAL = 20;
    cout << "Rolling: " << (rand() % MAX_VAL + 1)<< endl;
    cout << "Rolling: " << (rand() % MAX_VAL + 1)<< endl;
    cout << "Rolling: " << (rand() % MAX_VAL + 1)<< endl;
    cout << "Rolling: " << (rand() % MAX_VAL + 1)<< endl;
    // ...... some omitted here.....
    cout << "Rolling: " << (rand() % MAX_VAL + 1)<< endl;
    return 0;
}
```

```
Rolling: 17
Rolling: 11
Rolling: 17
Rolling: 17
Rolling: 2
Rolling: 2
Rolling: 18
Rolling: 8
Rolling: 7
Rolling: 8
Rolling: 19
Rolling: 6
```

Summary

- Keyboard input: cin >> var1;
- Chaining: cout << a << b; or cin >> x >> y;
- Expressions calculate values using operators.
  - Operator precedence gives us expression trees.
  - Implicit type conversions happen automatically.
  - Explicit type conversions by static_cast.
- Use named constants (const), not magic numbers.
- Combined assignment operators like x += 2;
- Math functions like pow(), ceil()
- Random functions srand(), rand(), and timer()