CS 161
Intro to CS I

Operators and Decisions
Odds and Ends...

• Assignment #2 due Sunday, 11:59pm
• Recitation Quiz #2 emailed Monday, 11:59pm
Constants

• What is a constant?

• How do we define a constant?
  – Use of a macro
    • `#define`
    • Placed at top of program
    • No semicolon at end
    • Example: `#define MAX_SIZE 100`

  – Use of `const`
    • Same as declaring variable but const
    • Example: `const int MAX_SIZE = 100;`
Intro to Macros

• C++: <climits>
• Use MIN and MAX macros from library
  http://www.cplusplus.com/reference/clibrary/climits/  
  (Note that the values listed are not the values on our system!!!)
  – INT_MAX
  – INT_MIN
  – LONG_MAX
  – LONG_MIN
  – SHRT_MAX
  – SHRT_MIN
• Remember unsigned too...
Expressions

• What is an expression?
  – Set of operations producing a value
    • Combining literal values
      12 * 4 + 6 * 10 vs. ((12 * 4) + 6) * 10
    • Combining variables
      var1 * var2 + var3 * var4 vs. ((var1 * var2) + var3) * var4
Expressions cont.

• Pieces of an Expression:
  – Operators
    • Indicate operation, e.g. +, *, /, -, %
  – Operands
    • Values in the expression
  – Evaluation
    • Process of obtaining results from operations on operands
Arithmetic Operators

- **Add**
  - 34 + 23
- **Subtract**
  - 34 - 23
- **Multiply**
  - 2 * 23
- **Divide**
  - 40 / 10
- **Remainder/Mod**
  - 34 % 5

The result of $34 \div 5$ is 6, with a remainder of 4. This means 4 is the remainder when 34 is divided by 5. If both operands are integers, the division $34 \div 5$ results in 6, with a remainder of 4.
Arithmetic

• Integer Arithmetic
  
  ```
  std::cout << 3/8;   /*prints 0*/
  std::cout << 34/5;   /*prints 6*/
  int age=5;
  std::cout << age/2;   /*prints 2*/
  ```

• Floating Point Arithmetic
  
  ```
  std::cout << 34.0/5.0;   /*prints 6.8*/
  std::cout << 3.0/8;   /*prints .375*/
  float years=2.0;
  std::cout << age/years;   /*prints 2.5*/
  ```
Type Casting

• Casting
  
  ```
  std::cout << age / (int) years; /*prints 2*/
  std::cout << (int) (age / years); /*prints 2*/
  std::cout << (float) age / 2; /*prints 2.5*/
  ```

• What is wrong with these?
  
  ```
  std::cout << (int) age / years; /*prints 2.5*/
  std::cout << (float) (age/2); /*prints 2.0*/
  ```
Precedence

• What is precedence?
  – Binding power of operator
  – (*, /, %) vs. (+, -)

• How do we override precedence?
  – Parenthesis!

• Examples:
  
  $12 \times 4 + 6 \times 10$ vs. $((12 \times 4) + 6) \times 10$

OSU Oregon State University
Additional Operators

• Common operation: fetch/store same variable
  
  ```
  var = var + 2;  // increment variable contents
  var = var * 2;  // double variable contents
  ```

  – Assignment/operator combination (all ops supported):
    ```
    var += 2;
    var *= 2;
    ```

• Pre/Post increment/decrement: ++ and –
  
  – Example: age++ vs. ++age
```cpp
#include <iostream>
#define BITS_BYTE 8

using namespace std;

int main() {
    int age=21;

    cout << "Where age is: " << &age
         << " Value is: " << age << endl;

    cout << age++ << endl; //post increment, print value then update
    cout << ++age << endl; //pre increment, update then print value

    //21=age; bad lvalue
    BITS_BYTE;

    return 0;
}
```
Decisions in Life

• What is a decision?
• When do we make decisions?
• How do we make decisions?
  
  If it is sunny today
  then I’ll go to the beach and fly a kite
  Else if it is raining today
  then I’ll stay inside and read a book
  Else if it is snowing
  then I’ll go to the mountains to ski
Decisions within Decisions

• What happens if there is no wind at the beach?

• How does this change our decisions?
  If it is sunny today
  
  then I’ll go to the beach

  if it is windy at the beach
  then I’ll fly a kite

  if it is not windy at the beach
  then I’ll walk on the shore
Flow chart for decisions

1. Is it sunny? Yes → Go to beach
   No → Is it raining? Yes → Read book
       No → Go outside

2. Is it windy? Yes → Fly kite
    No → Walk on beach
Decisions in our programs

- Use an if/else
  
  ```
  if (<expression>) {
      <statement>;
      ...
      <statement>;
  }
  else {
      <statement>;
      ...
  }
  ```
What is the `<expression>`?

Could be a relational expression:

```latex
<expression> <relational op> <expression>
```

- **Relational Ops**
  - `==` - equal to
  - `!=` - not equal to
  - `<` - less than
  - `>` - greater than
  - `<=` - less than or equal to
  - `>=` - greater than or equal to

\[ 1 == 1 \text{ true} \]
Examples

• if(2 + 1) //expression
  true
• if(2 – 4) //expression
  true
• if(2 – 2) //expression
  false
• if(4 == 4) //expression relational op expression
  true
• if((2+1) == 4) //expression relational op expression
  true
• if(4.1 != 4) //expression relational op expression
  true
• if(3 <= 4) //...
• if(4 >= 4)
• if(3.5 > 4)
• if(4 < 4)
• if(3+2*2 > 9)
• if((3+2)*2 > 9)
Logical Operators

• AND: if((1>2) && (2<5))
• OR: if((1>2) || (2<5))
• NOT: if(!(1>2) && (2<5))

• Precedence of Operators: refer to book
C++ If/Else Syntax...

```cpp
if( x > y) {
    cout << "X is greater than Y" << endl;
}
else {
    cout << "X is less than Y" << endl;
}
```

• When does this logic fail?
C++ If/Else...

```cpp
if( x > y) {
    cout << "X is greater than Y" << endl;
}
else if( x < y) {
    cout << "X is less than Y" << endl;
}
else {
    cout << "X is equal to Y" << endl;
}
```