Expressions and Operators

9-10-2001

Opening Discussion

- What did we talk about last class?
- Working in Linux out of class time. This room is open quite a bit of the time. I would be more than happy to have a "session" working on Linux and vi with people in here some afternoon.
- What is a variable?

Variables

- A variable in a programming language is nothing more than a place to store a value. Think of it as a box if you want.
- Variables in programming languages are similar to those in algebra, but they differ in very significant ways. Because of this you can put things in a program that would make no sense in algebra.
  - `n=n+1;`

Declarations

- In the C++ programming language, all entities must be declared before they are used. This is true for variables as well as other entities like functions and classes that we will discuss later.
- A variable declaration is simply a statement with a **type** and a name. Optionally it can also give it a default value. We'll talk more about types later today.

Assignments

- The way you store something in a variable is through an assignment.
- In C++ an assignment is denoted with an equal sign. What this actually indicates is that the value of the expression on the right side (rvalue) should be stored in the memory location of the variable on the left side (lvalue).
- Later we will talk about tests for equality.

Numeric Expressions

- Perhaps the most straightforward type of expressions in C++ are numeric expressions. These look very much like normal math formulae but you have to be a bit more rigid with how things are entered.
- Simple expressions can be literals (5, 2.9, etc.) or variables. Complex expressions can be formed by joining expressions with numeric operators.

Numeric Operators

- C++ has a fair number of numeric operators.
  - Common two argument operators (+, -, *, /)
  - Common one argument operator (-)
  - Less common two argument operators
    - % - remainder (or modulo)
    - & - bitwise and
    - | - bitwise or
    - ^ - bitwise xor (NOT exponent)
    - << and >> - left-shift and right-shift
  - Unary bitwise negation (~), not even in book.

Order of Operation

- For most of the normal operators the precedence is the same as in normal arithmetic (PEMDAS). You book lists the precedence of most all of the C++ operators in appendix C.
- It is generally recommended that you use parenthesis to make it clear how things are to be evaluated in an expression if there is any possible reason to question.

Types in C++

- A very significant feature of any programming language is it's typing system. Type here does not relate to a keyboard, but instead to categories of how the computer interprets what is stored at different memory locations.
- In C++ the type of a variable or object is very important as it determines what you can do with the object. For this class we will only use the strengths of the C++ typing system and not focus on the holes is has left over from C.

Integer Types

- There are a number of primitive types in C++. One collection of these are the integer types. These are types that represent whole number values.
  - int - This is what you will generally use.
  - short
  - long
  - char - used to represent characters too.
- Exact size of all depends on hardware and implementation. All can be modified with unsigned.

Floating Point Types

- When you want to represent a number that has a fractional part you use a floating point type
  - float - standard precision floating point number.
  - double - double precision floating point number. This is what you will use most of the time.
- These numbers are stored in computers much like scientific notation.

Polymorphism, Overloading, and Type Conversion

- In C++ the basic numeric operators work on both integers and floating point values. When a piece of code works with multiple types it is called polymorphism. In this case the polymorphism is created by overloading, one of the weakest forms of polymorphism.
- C++ also does implicit type conversions on numbers when types are mixed. It converts to the type that typically doesn't lose information.
  - 13.7 + 4 is a double plus an int. When evaluated it has the type double. Same for 13.0 + 4.

The string Type

- Another type that I used in the example code on Friday was the strong type. The numeric types just discussed are primitive types. They are typically "built into" the hardware I some way. They aren't really objects. Their only "attribute" is their value and they have no "behaviors". A string is a true object (an instantiation of a class). It allows you to store sequences of characters.
- Strings can be concatenated with '+', another instance of overloading.

Minute Essay

- Numerical expressions in C++ are quite similar to those you are used to from algebra with a few differences. Those differences though are significant and can cause problems with your comprehension of this topic. Do you feel comfortable with this now? Could you write a simple program that performs numerical operation on paper?