Programming Concepts
Basic Data Types

Fundamentals of Data Storage

- **Variables** are named storage locations where data is stored, which may be changed as a program runs. E.g. "nStudents".
- **Constants** are values that are hard-coded into a program, and which do not chnage value. E.g. "3.14159".
- Ultimately all data stored on a computer, both variables and constants, is stored as a sequence of binary digits, e.g. strings of zeros and ones.
  - These binary digits are referred to as "bits".
  - Physically these zeros and ones may be implemented using wires with two different voltages, magnetic particles with two different alignments, spots on an optical disk having two different optical properties, or by other means.
- The **"type"** of a particular variable or constant determines how many bits are used used for that paticular data item, and how the bits are to be interpreted.
- Most modern computer languages recognize five basic categories of data types: Integral, Floating Point, Character, Character String, and composite types, with various specific subtypes defined within each broad category. Most modern languages also include a number of language-specific special types, and many languages provide a means for programmers to define their own data types, and the operations that can be performed on them.
- Data may be converted from one type to another, (possibly with loss of precision), either automatically or specifically.

Integral Types

- Integral data types include all whole numbers, that is numbers not having any fractional component.
- The bits of integral types are interpreted as simple powers of two:
  - The right-most bit, known as the **least significant bit**, represents the number of 1s. (2^0)
  - The next bit represents the number of 2s. (2^1)
  - The next bit represents the number of 4s. (2^2)
  - The next bit represents the number of 8s. (2^3)
  - In general the nth bit from the right represents 2^(n-1)
- For unsigned integral types, the leftmost bit, known as the **most significant bit**, represents 2^(N-1), where N is the total number of bits in the data item.
  - The range of possible values for an unsigned integer of N bits is from 0 to 2^N - 1. (All 0s to all 1s)
  - So for example, a 4-bit unsigned integer could range from 0 to 15, and an 8-bit unsigned integer could range from 0 to 255.
- For signed integral types, the leftmost bit can be thought of as representing a **negative** 2^(N-1).
  - (The real interpretation in the computer is more complicated, but if you think of it this way you will get the right answers.)
  - The most negative value would be the first bit a 1 and all other bits 0s, yielding negative 2^(N-1). (E.g 1000 or 10000000)
  - The most positive value would be the first bit a 0 and all other bits 1s, yielding 2^(N-1) - 1. (E.g. 0111 or 01111111)
  - So for example, a 4-bit signed integer could range from -8 to +7, and an 8-bit signed integer could range from -128 to +127.

  - A signed integral type having all bits 1 is equal to -1, regardless of how many bits are in the number.
- Signed and unsigned integers with the same number of total bits have the same number of different possible values.
  - Unsigned integers use one bit pattern (all 0s) to represent zero and all others to represent positive values.
  - Signed integers use half of the possible bit patterns to represent negative numbers, one pattern to represent zero, and half minus 1 to represent positive values.
- Particular languages define different integral types based on the number of bits stored and whether or not the bits are interpreted as a signed or unsigned number. Some languages may reserve particular bit patterns to represent special values. (E.g. Matlab defines bit patterns to represent positive and negative infinity, as well as "NaN", i.e. "Not a Number")

Floating Point Types

- Floating point types include all types in which a number may have a fractional component, such as 42.7 or pi.
- Most modern computer languages represent floating point numbers using the IEEE floating point standard. Under this standard:
  - One bit is used as a sign bit, to indicate the sign of the number.
  - A certain specified number of bits are used to represent the numeric digits of the number, such as 314159 or 602.
  - The remainder of the bits are used to indicate where the decimal point (or binary point) is located within the digits, such as -5 to yield 3.14159 or 21 to yield 6.02 * 10^23.
  - A few bit patterns are reserved to represent plus and minus infinity, plus and minus zero, and two kinds of NaN. (Not all computer languages support these special types.)
- Different floating point data types differ by their ***precision*****,** determined by the number of bits used to represent digits, which is in turn directly related to he total number of bits in the number. For example in the IEEE standard:
  - ***Single precison*** floats use 32 total bits and 24 bits for digits, yielding about 7 decimal digits of precision and a range from about 10^-38 to 10^38.
  - ***Double precison*** floats, often just called ***doubles***, use 64 total bits and 53 bits for digits, yielding about 16 decimal digits of precision and a range from about 10^-308 to 10^308.
  - ***Quadruple precison*** floats, often called long doubles quad doubles, or double doubles, use 128 total bits and 113 bits for digits, yielding about 34 decimal digits of precision and a range from about 10^-4932 to 10^4932.
- Regardless of the precision, all floating point numbers should be considered as rounded-off approximations in digital computers. For example, the decimal value 0.1, i.e. 1/10, cannot be represented as an exact floating-point number using binary digits as described above.

Characters

- Most modern compter languages store alphabetic characters by reprsenting each one using a small numeric code.
- Because the number of characters that must be stored is typically a large number, and the number of bits needed to represent each character uniquely is small, characters are typically stored as 8-bit unsigned integers, having a range of codes from 0 to 255.
  - 8 Bits in a computer is termed a ***byte.*** Another common name for this very small integer type is ***char.***
  - The most common encoding of 8-bit chars is the ASCII code, which can be found in any good programming bookor online at http://www.asciitable.com/ and many other sites.

        - For example, the ASCII code for the capital letter 'A' is 65.
    - Note that the character '9' is not the same as the integer value 9.
        - In this case the ASCII code for the character '9' is 57.
        - The numerical value 9 in the ASCII code set happens to represent a horizontal tab character.
- Although the char data type is most commonly used for holding codes representing alphabetic characters, it is technically a small integer, and many languages allow chars to be used and manipulated like any other integer. (E.g. multiplication, addition, etc.)
- In order to support a wide variety of special symbols, as might be found in foreign languages, mathematics, science, character graphics, or elsewhere, Java and some other languages support ***wide characters,*** (wchars), having 16 bits instead of 8.
    - It has been said that every known language, both current and historical, can be reprsented by 16-bit Unicodes.

Character Strings

- Most modern computer languages have some means of representing character strings, i.e. collections of characters such as words or sentences, as well as a library of methods for manipulating those strings.
- For example:
    - Traditional C stores character strings as ***arrays*** of characters, terminated by a null byte. They are also commonly manipulated using ***pointers*** to characters.
    - Both C++ and Java define String ***classes***.

Pointers

- Pointers are variables that hold the ***address*** of where some other data or code is stored.
- Pointers are used to indirectly access the item(s) pointed to by the pointers.
- Pointer variables can be changed, so as to point to other locations.

Composite Data Types

Composite data types are collections or groupings of multiple data items into a single entity. Different languages provide support for different kinds of composite data types.

Arrays

- Arrays are collections of data items all having the same data type, accessed using a common name and an integer index.

- In theory arrays can have any (positive) number of dimensions, using one additional index for each new dimension. In practice most real languages impose some kind of limit on array dimensions, although the limit is generally much higher than a well-written program should ever use.
- In Matlab the array is the basic data type, so even a single number is considered as a 1 x 1 array.
- In C and C++ arrays are accessed indirectly, using pointer arithmetic relative to the address where the beginning of the arrray is stored.

Structures

- Structures, (called ***structs*** in C/C++), are collections of data items of either the same or different types, each having a unique name within the collection, and accessed using those names.
- For example, a structure representing a Book might have String variables for the title and author, integer variables for the current page number and the total number of pages, and a floating point value for its cost.

Unions

- Unions are very much like structures, except that they only reserve enough space for the largest contained data item, which is then shared by all of the defined fields within the union.

Classes

- Classes are also very much like structures, except that in addition to data, they also contain the functions (methods) associated with the data. Classes can be viewed in two different but equally valuable ways:
  - A collection of related data representing an entity, plus the associated methods needed to maintain and manipulate that data.
  - A collection of related services, (methods, functions), plus the associated data necessary to deliver those services.
- The concept of bundling together a group of related data items which collectively represent a single entity is termed ***encapsulation.***
- Classes also typically implement ***information hiding,*** allowing external code to use the services provided by the classes without having knowledge of or access to the internal working of the classes.

Programmer-Defined Types

Special Types

- The **enumerated** type is an integer with a restricted list of legal values, referred to by names. It will be covered in full details in the section on structs and unions.
- C99 introduces new types _Bool, _Complex, and _Imaginary.

Enumerated Types (Advanced, Optional)

- Enumerated (enum) data types are basically ints, except that they are restricted to a limited set of values, and those values are referred to by name not by number.
- The use of enums where applicable helps make code more readable and also limits the possibilities for bad values, thereby reducing bugs and making the code more maintainable and overall better.
- The enum keyword is used to define a new data type, having a new data type name and list of acceptable named values.
- Once the new enum type has been declared, variables can be declared of the new type, and assigned the named values.
- For example:

```
enum SizeType { small, medium, large };  // Declares a new data type, "SizeType"

SizeType item;    // Declares a variable of type "SizeType"

// ( Some code left out here. )

if( num < 25 )
    item = small;  // Use as an int, using the named values instead of numbers

cout << "\nThe item is ";

switch( item ) {

    case small:              // Named values are valid integers
        cout << "tiny\n";
        break;
```

- Named values can be assigned specific numbers. Those not assigned will get successive values. So in the following example, minor2 will have the value 2 and major2 will have the value 101:

```
enum errorType { none = 0, minor1 = 1, minor2, major1 = 100, major2, fatal1 = 1000 };
```

- Enumerated type variables can also be initialized.. For example:

```
errorType errorCode = none;
sizeType bookSize = large;
```

- It is sometimes a good idea to include values such as "invalid", "undefined" or "none" among the list of enumerated values.
- Some compilers may allow using enum variables with ordinary integers, (e.g. using numbers instead of names), but it is poor practice.
- Printing enumerated variables prints the assigned integer value.
- One should not attempt to do any math using enumerated variables.

Related Topics

The following topics are not covered here, but may be found in many books on C/C++;

- User-Defined Types (typedef)
- Enumerated Types (enum)