Simple Data Types: Built-In and User-Defined

- Additional C++ operators: +=, -=, *=, /=

Statement	Equivalent Statement
`i += 5;`	`i = i + 5;`
`pivotPoint *= n + 3;`	`pivotPoint * (n + 3);`

- The Cast Operation

  Two forms:

  intVar = (int) floatVar;

  intVar = int(floatVar); (only works if the data type name is a single identifier)

  myVar = (unsigned int) someFloat; (the only way for this one)

Working with Character Data

- Each computer uses a particular character set, two in substantial use today are ASCII and EBCDIC. The vast majority use ASCII.
- Character Issues:
  - Comparing Characters: if (ch >= ‘a’ && ch <= ‘z’) – islower is preferred usage
  - Converting Digit Characters to Integers: ‘9’ – ‘0’ = 9
  - Converting to Lowercase and Uppercase: toupper and tolower can help (#include <cctype>)
  - Accessing Characters Within a String: Use position number, e.g. ch = inputStr[2];

More on Floating-Point Numbers

- Comparing Floating-Point Numbers: check for an error within some range
- Underflow and Overflow: Numbers can be too large or too small to be represented.
- Cancellation Error: problems can occur when arithmetic operations occur on values of wildly different magnitudes: (1 + 0.00001234 – 1) = 0.00001234. For four digits of precision, this doesn’t work, why?

Practical Implications of Limited Precision

- Limited precision has led to
  - Mercury splashdown in unknown location, delaying the recovery of the spacecraft and the astronaut.
  - Specialized high-tension cables linking a hydroelectric dam to the nearest power distribution point had to be redone due to precision problems.
  - A bank employee stole infinitesimal amounts from a large number of accounts.

User-Defined Simple Types

- Enumeration Types – C++ allows the user to define a new simple type by listing (enumerating) the literal values that make up the domain of the type. These literal values are identifiers, not numbers, separated by commas.
- Example:

```
enum Days (SUN, MON, TUE, WED,
  THU, FRI, SAT);
```


Enumeration Type Declaration

- EnumDeclaration

  enum Name {Enumerator, Enumerator, …};
- Enumerator

  Identifier = ConstIntExpression
- Legal or Illegal?

```
enum Vowel {‘A’, ‘E’, ‘I’, ‘O’, ‘U’};
enum Places {1st, 2nd, 3rd};
enum Starch {CORN, RICE, POTATO, BEAN};
enum Grain {WHEAT, CORN, RYE, BARLEY, SORGHUM};
```

More About Enumeration Types

Consider the following declaration:

```
enum Animals {RODENT, CAT, DOG, BIRD, REPTILE, HORSE,
BOVINE, SHEEP};
```

and the following variable declarations:

```
Animals inPatient;
Animals outPatient;
```

Acceptable assignemnts are

```
inPatient = DOG;
outPatient = inPatient;
someInt = DOG;
inPatient = Animals(inPatient + 1);
```


Typical Usage of an Enumeration Type

```
switch (inPatient)
{
   case RODENT   :
   case CAT      :
   case DOG      :
   case BIRD     : cout << "Cage ward";
                   break;
   case REPTILE  : cout << "Terrarium ward";
                   break;
   case HORSE    :
   case BOVINE   :
   case SHEEP    : cout "Barn";
                   break;
}
```


Enumeration Type Input

- Enumeration types are known only internally to the program, they are input and output indirectly.

```
string animalName;
cin >> animalName;

switch (toupper(animalName[0]))
{
   case 'R' : if (toupper(animalName[1] == 'O')
                 inPatient = RODENT;
              else
                 inPatient = REPTILE;
              break;
   case 'C' : inpatient = CAT;
              break;
               :
```


Enumeration Type Output

- Enumeration types are known only internally to the program, they are input and output indirectly.

```
switch (inPatient)
{
   case RODENT :  cout << "Rodent";
                  break;
   case CAT    :  cout << "Cat";
                  break;
   case DOG    :  cout << "Dog";
                  break;
   case BIRD   :  cout << "Bird";
                  break;
               :
}
```


Using a Value-Returning Function for Input

```
Animals StrToAnimal (/*in*/ string str)
{
  Animals inPatient;
  switch (toupper(str[0]))
  {
     case 'R' : if (toupper(animalName[1] == 'O')
                   inPatient = RODENT;
                else
                   inPatient = REPTILE;
                break;
     case 'C' : inPatient = CAT;
                break;
     case 'D' : inPatient = DOG;
                break;
                 :
                 :
     return inPatient;
  }
}
```


User-Written Header Files

- User-defined data types can be useful in more than one program.
- Declarations can be placed in a header file and included using the #include directive.
- #include <iostream> looks in the standard include directory
- #include "months.h" looks in the current directory