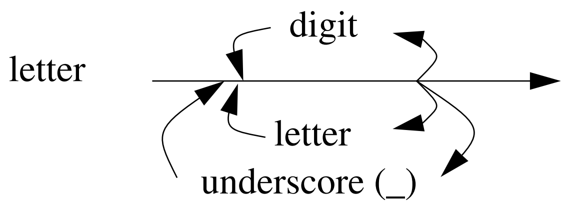
C++ Overview

- Reserved Keywords
    - o special meaning
    - o list of reserved keywords:

      asm, auto, break, case, catch, char, class, const, continue, default, delete, do, double, else, enum, extern, float, for, friend, goto, if, inline, int, long, new, operator, private, protected, public, register, return, short, signed, sizeof, static, struct, switch, template, this, throw, try, typedef, union, unsigned, virtual, void, volatile, while

- Identifier
    - o a name
    - o a letter followed by any sequence of letters, digits or underscores



- ***o case-sensitive***
- o e.g. Hello, Id200, WhoAmI, an_identifier, •••
- o Invalid identifiers: X=Y, J-20, 20JJ, •••
- o **cannot** use a C++ reserved keyword as an identifier

C++ Overview (Cont'd)

- Comment
  - o explanatory notes, not part of the program
  - o two forms:

```
// A double slash marks the start of a single line comment

/* A slash followed by an asterisk mark the beginning of a
   multiple line comment.
   It  ends with an asterisk followed by a slash          */
```

- Compiler Directive (#include)
  - o header file for using library function or variables
  - o reads in the contents of the named file
  - o preprocess (before program is compiled)
  - o two forms:

```
#include    <iostream.h>
#include    "my_lib.h"
```

- Compiler Directive (#define, #ifdef, #endif)
  - o e.g.

```
#define   DEBUG
•••
#ifdef   DEBUG
// DEBUG flag set.   Print out the following variables:
•••
#endif
•••
```

C++ Overview (Cont'd)

- Constant Declaration
    - o associate specific value with a constant identifier
    - o cannot be changed at any time by the program
    - o e.g. **const** float pi = 3.14159;

- Variable Declaration
    - o allocate storage for variable
    - o e.g. **int** v1, v2;
      float v3, v4;

- Executable statements (assignment statement)
    - o e.g. v1 = 2 + 1;
    - o variable v1 is assigned a value of 3
    - o previous value of v1 is destroyed

- General form of a C++ program

```
// Program Description

#include directives

void main(){

  constant declarations
  variable declarations

  executable statements
}
```

Example

- Example program:

```
// Convert inches to feet and inches
// Input: inches
// Output: feet and inches

#include <iostream.h>

void main() {

     /* inches to feet conversion factor */
     const int in2feet = 12;

     int  inches; /* number of inches */

     cout<< "Enter number in inches: ";
     cin >> inches;

     cout<< inches / in2feet << " feet "
         << inches % in2feet << " inches "
         << endl;
}
```

Data Types

- a set of ***values***
- a set of ***operations*** on those values

int Data Type

- represent ***integer*** (e.g. -32768 to 32767)
- operators (+, -, *, / [quotient] , % [remainder])
  - o 15 / 2 = 7
  - o 49 % 5 = 4

float Data Type

- represent values with a decimal point and fractional part
- operators (+, -, *, /)

Internal Representation

- All variable values are represented by binary bits.
- Different machine has different word size
  - 16-bit word, 32-bit word, 64-bit word, etc.
- The language C++ does not define the actual representation
  - Machine and compiler dependent

Integer:

sign	binary number

Real number:

sign	exponent	mantissa

Variables

Variables is best thought of as a container for a value.

1. A variable must be declared before it can be used.

2. A variable must be assigned some value (initialized) before its content can be referenced, i.e., must be defined first. Do not assume that the variable will be initialized.

3. A variable must have only one type and it can contain only values of that type. For example, a variable of the type **int** can only hold integer values.

4. Once a value has been placed in a variable it stays there until the program deliberately alters it.

Constant

Variables hold values that change.

Constant represent values permanently

Make program more readable & maintainable

Declaration:

const <type> <identifier> = <constant expression>;

Example:

```
const int Two = 2;

const int Four = Two * 2;
```

Variable Declaration

<type> <identifier>, <identifier>, ••• ;

For example,

```
int     integer1, integer2;
float   f1, f2, f3;
```

Assignments and Expressions

< identifier> = <expression>;

Examples:

```
int  I, J, K;
I = 5;
J = 6 + 4 * 6;
K = I * 2 + J;
K = K / 2;
```

Expressions (cont'd)

Operator precedence:

*   /   %   is completed before

+   -

Examples:

4 + 2 * 3

5 * 3 + 14 % 4

49 / 8 % 4

Parentheses:

- change the order of evaluation
- inside out
- recommended to make expression clear (even not required)

Examples:

(49 / 8) % 4

Mixed-Type Assignment

Example:

```
int      M, N;
float    X, Y;
M = 3;
N = 2.5;      // 2.5 is converted to 2 and assigned toN
X = M / N;  // 3/2=1 is converted to 1.0 and assigned to X }


N = X + M / 2;
// M / 2 = 1 : integer division
// X + M / 2 : float addition because X is of float type
// convert result of M / 2 to float type (i.e. 1.0)
// X + M / 2 = 2.0
// convert result of X + M / 2 to int type (i.e. 2)
//      because N is of int type
```

Standard Input/Output

- cin - the standard input stream

  Input operator ">>"

  - Extracting data from input stream
  - Skip white spaces
  - Extract only characters of the right form and perfrom automatic conversion

- cout - the standard output stream

  Output operator "<<"

  - Insert data into output stream

- Example:

```
int id, score;

cout << "Enter student ID and score: ";
cin >> id >> score;

cout << "Student ID: " << id
     << " score: " << score
     << endl;
```