Chapter 3
Expressions and Interactivity

The \texttt{cin} Object

- We’re using an object but not creating new objects (object-oriented programming).

- user input goes from keyboard to program
- converts information to the type of the variable:
  \begin{verbatim}
  int height;
  cout << "How tall is the room? ";
  cin >> height;
  \end{verbatim}

\begin{verbatim}
user keys the value and presses <enter>
\end{verbatim}

The \texttt{cin} Object

- can be used to input one or more values:
  \begin{verbatim}
  cin >> height;
  cin >> width;
  or
  cin >> height >> width;
  \end{verbatim}

- user must separate multiple values by spaces (not commas)
- order is important: first value entered goes to first variable, etc.

The \texttt{cin} Object

- can be used with \texttt{cin} to assign a value:
  \begin{verbatim}
  char myName[20];
  cin >> myName;
  \end{verbatim}

- stops input on whitespace
  \begin{verbatim}
  <space bar> <enter> <tab>
  \end{verbatim}

- key: John Doe
  myName receives John only
  space stops the input
  Doe remains in the input buffer

The \texttt{cin} Object

- can be used to read in a string
- must first declare an array to hold characters in string:
  \begin{verbatim}
  char myName[20];
  \end{verbatim}

- myName is name of array
- 20 is the number of characters that can be stored
- 20 must include the NULL character at the end so you could only store 19 characters of the name

The \texttt{cin} Object

- Standard input object
- Like \texttt{cout}, requires \texttt{iostream} file to be included
- Used to read input from keyboard
- Often used with \texttt{cout} to display a user prompt first
- Information retrieved from \texttt{cin} with >>
  (extraction operator)
- Input information stored in one or more variables
Mathematical Expressions

- An expression can be a constant, a variable, or a mathematical combination of constants and variables

```cpp
constPerHour = 78;
sqLength = sqWidth;
area = 2 * pi * radius;
cout << "border is: " << 2*(l+w);
```

Order of Operations

- ( ) innermost to outermost
- * / % in order, left to right
- + - in order, left to right

In the expression `2 + 2 * 2 – 2`,

Evaluate first

Evaluate second

Evaluate third

**Associativity of Operators**

- Parentheses ( ) can be used to override the order of operations:

  - `2 + 2 * 2 – 2`
  - `(2 + 2) * 2 – 2`
  - `2 + 2 * (2 – 2)`
  - `(2 + 2) * (2 – 2)`

**Evaluate the following given:**

```cpp
int inum;     float fnum;
```

- `inum = 19 / 5;`
- `inum = 19 % 5;`
- `fnum = 19.0 / 5.0;`
- `fnum = 19.0 / 5;`
- `fnum = 19 / 5;`
- `fnum = 3 + 3 * 5;`
- `inum = 3 + 2 – 4 + 6;`

---

**Evaluate the following given:**

```cpp
int inum;     float fnum;
```

- `inum = 6 + 4 / 2 + 3;`
- `inum = (6 + 4) / (2 + 3);`
- `inum = (2 * (3 + 7)) / 5;`
- `inum = 8 + 5 * 7 % 2 * 4;`
- `fnum = 10.0 + 15 / 2 + 4.3;`
- `fnum = 10.0 + 15 % 2 + 4.3;`
- `fnum = 24 % 5 + 7 * 4/7;`

---

**Evaluate the following given:**

```cpp
int inum;     float fnum;
```

- `inum = 19 / 5;`

- The value of `inum` is 3
Evaluate the following given:
int inum; float fnum;

- \text{inum} = 19 \% 5;

- The value of inum is 4

Evaluate the following given:
int inum; float fnum;

- \text{fnum} = 19.0 / 5.0;

- The value of fnum is 3.8

Evaluate the following given:
int inum; float fnum;

- \text{fnum} = 19.0 / 5;

- The value of fnum is 3.8

Evaluate the following given:
int inum; float fnum;

- \text{inum} = 19.0 / 5.0;

- The value of inum is 3

Evaluate the following given:
int inum; float fnum;

\text{inum} = 3 + 3 \times 5;

\begin{array}{c}
3 + 15 \\
\end{array}

The value of inum is 18
Evaluate the following given:
```c
int inum; float fnum;
inum = 3 + 2 - 4 + 6;
```
```
5 - 4
1 + 6
```
The value of inum is 7

Evaluate the following given:
```c
int inum; float fnum;
inum = 6 + 4 / 2 + 3;
```
```
6 + 2
8 + 3
```
The value of inum is 11

Evaluate the following given:
```c
int inum; float fnum;
inum = (6 + 4) / (2 + 3);
```
```
10 / 5
```
The value of inum is 2

Evaluate the following given:
```c
int inum; float fnum;
inum = (2 * (3 + 7)) / 5;
```
```
2 * 10
20 / 5
```
The value of inum is 4

Evaluate the following given:
```c
int inum; float fnum;
inum = 8 + 5 * 7 % 2 * 4;
```
```
35 % 2
1 * 4
8 + 4
```
The value of inum is 12

Evaluate the following given:
```c
int inum; float fnum;
fnum = 10.0 + 15 / 2 + 4.3;
```
```
17.0
```
The value of fnum is 21.3
Evaluate the following given:

```c
int inum;     float  fnum;
fnum = 10.0 + 15 % 2 + 4.3;
```

The value of fnum is 15.3

Evaluate the following given:

```c
int inum;     float  fnum;
fnum = 24 % 5 + 7 * 4/7;
```

The value of fnum is 8.0

### Algebraic Expressions

- Multiplication requires an operator: 
  \[ \text{area} = l \times w \]
  is written as \[ \text{area} = l * w; \]
- There is no exponentiation operator: 
  \[ \text{area} = s^2 \]
  is written as 
  \[ \text{area} = \text{pow}(s, 2); \quad \text{or} \quad \text{area} = s * s; \]
- Parentheses may be needed to maintain order of operations: 
  \[ m = \frac{y_2 - y_1}{x_2 - x_1} \]
  is written as 
  \[ m = (y_2 - y_1) \div (x_2 - x_1); \]

### Writing C/C++ statements from math:

- \[ y = a \times (b + c) \]
  \[ y = a \times (b + c); \]
- \[ x = \frac{a + 6}{b} \]
  \[ x = (a + 6) \div b; \]
- \[ \text{vol} = \frac{4}{3} \pi r^3 \]
  \[ \text{vol} = 4.0 \div 3.0 \times \pi \times r \times r \times r; \]
  \[ \text{vol} = 4.0 \div 3.0 \times 3.14 \times r \times r \times r; \]
  \[ \text{vol} = 4.0 \div 3.0 \times 3.14 \times \text{pow}(r, 3); \]
Mixing Data Types

Type Conversion

- Operations are performed between operands of the same type.
- If not of the same type, C++ will automatically convert one to be the type of the other for this operation only.
- This is called type coercion and results in promotion or demotion.
- **Type Coercion**: automatic conversion of an operand to another data type
- **Promotion**: convert to a higher type
- **Demotion**: convert to a lower type

Hierarchy of Types

<table>
<thead>
<tr>
<th>Highest</th>
<th>Lowest</th>
</tr>
</thead>
<tbody>
<tr>
<td>long double</td>
<td>short</td>
</tr>
<tr>
<td>double</td>
<td>int</td>
</tr>
<tr>
<td>float</td>
<td>unsigned int</td>
</tr>
<tr>
<td>unsigned long</td>
<td>long</td>
</tr>
</tbody>
</table>

Coercion Rules

- char, short, unsigned short automatically promoted to int
- the lower type is promoted to the higher type
- on assignments (=), the type of expression on right will be converted to type of variable on left at assignment time
- type of variable on the left does not change the processes on the right

```
int test1, test2, test3;
float average;
average = (test1 + test2 + test3) / 3;
```

Typecasts

```
#include <iostream>
using namespace std;

int main()
{
    int i=7, j=2;
    cout << i / j << endl;
    cout << (float) i / j << endl;
    cout << float ( i ) / j << endl;
    cout << static_cast<float> ( i / j ) << endl;
    return 0;
}
```

#define directives and named constants

```
#define PID 3.14
float area, radius;
area = PID * radius * radius;
const double Pi = 3.14;
float area, radius;
area = Pi * radius * radius;
```

Multiple Assignment

- = is an operator that can be used multiple times in an expression:
  - x = y = z = 5;
  - counter = accumulator = 0;

Remember: Variables are NOT automatically initialized to zero.
Combined Assignment or Arithmetic Assignment

- C++ shorthand for common mathematical operations:
  - sum = sum + newnum;
  - sum += newnum;
  - pay = pay * 1.1;
  - pay *= 1.1;
- Operators: +=, -=, *=, /=, %=

Formatting Output

- control how output displays for numeric, string data:
  - field size
  - position
  - number of digits
- used with print charts
- implemented in QBASIC with the PRINT USING
- requires iomanip header file

Stream Manipulators

- Some affect just the next value displayed:
  - setw(x): minimum field width of x
    use more spaces if field is not wide enough
- Some affect values until changed again:
  - fixed: use decimal notation for floating-point values
  - setprecision(x):
    When used with fixed, print x decimal digits.
    Without fixed, print x significant digits
  - showpoint: always print decimal for floating-point values

Manipulator Examples

```cpp
const float e = 2.718;
cout << setw(8) << e;   // 2.718 in a field 8 wide
// 2.718 in a field 8 wide
\Delta\Delta2.718
where \Delta means space

cout << setprecision(2);
cout << e;              // 2.7
significant digits

cout << fixed << setprecision(2);
cout << e;             // 2.72
decimal
digits
```

Formatted Input

- Useful when reading string data to be stored in a character array:
  - char name[10];
  - cout << "Enter your name: ":
  - cin >> setw(10) >> name;
- cin reads one fewer character than specified in setw()
  (needs space for NULL)
- Still stops on whitespace
- overlays other memory if size is exceeded

Keying: Christopher
Stores: Christoph

Formatted Input

- To read an entire line of input, use cin.getline():
  - char address[81];
  - cout << "Enter your address: ":
  - cin.getline(address, 81);
- does NOT stop on whitespace
- cin.getline takes two arguments:
  - Name of array to store string
  - Size of the array
Formatted Input

• To read a single character:
  – Use cin:
    char ch;
    cout << "Strike any key to continue";
    cin >> ch;
  Problem: will skip over blanks, tabs, <enter>
  – Use cin.get():
    char ch;
    cout << "Strike any key to continue";
    cin.get(ch);
    Will read the next character entered, even whitespace
  – Both are buffered (requires pressing <enter> key)
  – Mixing cin and cin.get() in the same program can cause
    input errors that are hard to detect

Formatted Input

• To skip over unneeded characters that are still
  in the keyboard buffer, use cin.ignore():
  cin.ignore();
  // skip next char
  cin.ignore(10, "\n");
  // skip the next 10 characters or until
  "\n"

More About Member Functions

• Member Function: procedure that is part of an object
• cout, cin are objects
• Some member functions of the cin object:
  – getline
  – get
  – ignore
• >> and << are overloaded operators

Mathematical Library Functions

• Many require cmath header file
  – #include <cmath>
• Take double as input, return a double
• Commonly used functions:
  sin  Sine
  cos  Cosine
  tan  Tangent
  sqrt Square root
  log  Natural (e) log
  abs  Absolute value (takes and returns an int

Mathematical Library Functions

• Some require cstdlib header file
  – #include <cstdlib>
• rand(): returns a random number (int) between 0 and
  the largest int the computer holds.
  Yields same sequence of numbers each time program
  is run.
• srand(x): initializes random number generator with
  unsigned int x

File Input and Output

• Can use files instead of keyboard and screen
• Allows data to be retained between program runs
• Steps:
  – Define/declare the file
  – Open the file
  – Use the file (read from, write to, or both)
  – Close the file
### Files: What is Needed

- Use fstream header file for file access
  - `#include <fstream>`
- File stream types:
  - `ifstream` for input from a file
  - `ofstream` for output to a file
  - `fstream` for both input from or output to a file
- Define file stream objects:
  - `ifstream infile;`
  - `ofstream outfile;`

### Opening Files

- Create a link between file name (outside the program) and file stream object (inside the program)
- Use the open member function:
  ```
  infile.open("inventory.dat");
  outfile.open("a:\lab2\report.txt");
  ```
- Filename may include drive, path info.
- Output file will be created if necessary; existing file will be erased first
- Input file must exist for open to work

### Using Files

- Can use output file object and `<<` to send data to a file:
  ```
  outfile << "Inventory report";
  ```
- Can use input file object and `>>` to copy data from file to variables:
  ```
  infile >> partNum;
  infile >> qtyInStock >> qtyOnOrder;
  ```

### Closing Files

- Use the close member function:
  ```
  infile.close();
  outfile.close();
  ```
- Don't wait for operating system to close files at program end:
  - may be limit on number of open files
  - may be buffered output data waiting to send to file