Example

```
#include < iostream.h >
#include < iomanip.h >

void main ( ) {
        int  id ;
        float  total,  average ;

        cout << "Enter id, total and average: ";
        cin  >>  id  >>  total  >>  average;

        cout << "01234567890123456789012345678 9";
        cout  <<  setiosflags (          ios :: fixed  |
                                         ios :: showpoint  |
                                         ios :: left  )
                <<  setprecision ( 2 )
                << setw ( 8 )  << id
                << setw ( 10 )  << total
                << setw ( 10 )  << average
                << endl;

        cout.setf ( ios :: fixed );
        cout.setf ( ios :: showpoint);
        cout.setf ( ios :: right );
        cout.precision ( 2 );
        cout.width ( 8 );
        cout << id;
        cout.width ( 10 );
        cout << total;
        cout.width ( 10 );
        cout << average  << end;
}
```

Output:

012345678901234567890123456789

Files

- ***file***
  - o stored on secondary storage device (e.g. disk)
  - o permanent
  - o program can input data from file and output data to file

- ***stream*** - a sequence of characters
  - ***o interactive (iostream.h)***
    - ***cin*** - input stream associated with ***keyboard***
    - ***cout*** - output stream associated with ***display***
  - ***o file (fstream.h)***
    - ***ifstream*** *ins* *(ins.open(in_file))*
      - * input stream *ins* associated with ***file*** *in_file*
    - ***ofstream*** *outs* *(outs.open(out_file))*
      - * output stream *outs* associated with ***file*** *out_file*

File Manipulation Functions

- ***o*** *xxx.* ***open (*** *fname* ***)***
  - connects iostream *xxx* to external file *fname*
- ***o*** *xxx.* ***get (*** *ch* ***)***
  - extracts next character from an input stream *xxx* and place it in character variable *ch*
- *o* *xxx.* ***put (*** *ch* ***)***
  - inserts character *ch* to output stream *xxx*
- *o* *xxx.* ***eof ( )***
  - tests for the end-of-file condition
- *o* *xxx.* ***fail ( )***
  - returns a nonzero value (true) if operation failed
- *o* *xxx.* ***close ( )***
  - disconnects the stream and associated file

Example

```
// Counts the number of space in each line in a file
#include <iostream.h>
#include <fstream.h>


// External file names
#define in_file "in_data.dat"



void main() {
     const char blank = ' ';  // space
     const char nwln = '\n';  // newline character


     ifstream ins;  // ins is input stream


     int blank_count = 0; // no. of blanks in file
     char next;           // next character

     // open input file, exit on error
     ins.open (in_file);
     if (ins.fail ()) {
          cerr << "***ERROR: Cannot open "
               << in_file << endl;
           return;
      }
```

```
    ins.get(next);  // get first character

    while (!ins.eof()) {

        blank_count = 0;

        while ((next != nwln) && !ins.eof()) {
            cout.put(next);    // echo characters
            if (next == blank) // count blank
                blank_count++;
            ins.get(next);  // get next character
                                // of current line
        }

        cout.put(nwln);   // end of this line

        cout << "No. of blanks = "
             << blank_count << endl;
        if ( !ins.eof() )
            ins.get(next);     // get first character
                             //   of next line
    }

    ins.close();   // close input file stream
    return;
}
```

```
// Copy file in_data.dat to file out_data.dat
#include <iostream>
#include <stdlib.h>
#include <fstream.h>

#define in_file "in_data.dat"
#define out_file "out_data.dat"

void copy_line(ifstream&, ofstream&);

int main() {
   int line_count;
   ifstream ins;
   ofstream outs;

   ins.open (in_file);
   if (ins.fail()) {
      cerr << "***Error: Can't open input file- "
            << in_file << endl;
      return EXIT_FAILURE;
   }

   outs.open (out_file);
   if (outs.fail()) {
      cerr << "***Error: Can't open output file- "
            << out_file << endl;
      return EXIT_FAILURE;
   }

   line_count = 0;

   do {
      copy_line (ins, outs);
      line_count++;
   } while ( ! ins.eof() );

   cout << "Number of lines copied: " << line_count
         << endl;

   ins.close();
   outs.close();
   return 0;
}
```

```
// Copy one line of text from one file to another
void copy_line(ifstream& ins, ofstream& outs) {

// Precond: ins is opened for input and outs for
//             output
// Postcond: Next line of ins is written to outs.
//             The last character processed from
//             ins is <nwln>.
//             The last character written to outs
//             is <nwln>.

   const char nwln = '\n';
   char next_ch;

   ins.get (next_ch);
   while ((next_ch != nwln) && ! ins.eof()) {
      outs.put (next_ch);
      ins.get (next_ch);
   }

   // If last character read was nwln write it to
   // out_file.
   if ( ! ins.eof() )
      outs.put(nwln);

   return;
}
```

Enumeration Types

- **Motivation**

```
Example 1:

  if (DayNum == 1)
    PayFactor = 2.0;
  else if (DayNum == 7)
    PayFactor = 1.5;
  else
    PayFactor = 1.0;


Example 2:

  if (Today == Sunday)
    PayFactor = 2.0;
  else if (Today == Saturday)
    PayFactor = 1.5;
  else
    PayFactor = 1.0;
```

- Enumeration types make programs easier to read and understand

Enumeration Types (Cont'd)

- enumeration types provide abstraction through use of human-oriented terms
- each enumerator is a ***constant***, type ***int*** identifier
- unless explicitly indicated (<identifier = constant expression>) enumerator values start at **0** and increase in step of one
- unspecified enumerator values continue in steps of one from the last specified value
- Syntax

```
enum <type name> {<identifier>,•••,
                         identifier>};
```

- Examples:

```
enum PrimColor {Red, Green, Blue};
                 0     1      2

enum WeekDay {Mon,Tue,Wed,Thu,Fri,Sat,Sun};
              0   1   2   3   4   5   6

enum FirstQuarter {Jan=1, Feb, Mar};
                    1      2    3

enum logical {false, true}
               0      1

enum example {a, b=3, c, d, e=1, f, g}
              1  3    4  5  1    2  3
```

Example:

```
#include <instream.h>

void main ( ) {
   const  int rate = 50.0;
   enum  Day = {Sunday, Monday, Tuesday, Wednesday,
                        Thursday, Friday, Saturday};
   Day today;
   float PayFactor, hours;

       cout << "Enter number of hours worked: ";
       cin >>  hours ;
       cout << "What day is today? ";
       cin >> today;

       if  ( Today == Sunday )
           PayFactor = 2.0;
       else if  ( Today = Saturday )
           PayFactor = 1.5;
       else
           PayFactor = 1.0;

       cout << "Your pay is $"
              << PayFactor*rate*hours << endl;
}
```

Enumeration Types (Cont'd)

- enumeration type value has **NO** external character representation

```
enum PrimColor {Red, Green, Blue};
PrimColor color;

color = Green;
cout << color;

1
```

- Example: (Output for enumerated types)

```
enum PrimColor {Red, Green, Blue} Color;

Color = Green;

switch (Color) {
  case Red:   cout << "Red";   break;
  case Green: cout << "Green"; break;
  case Blue:  cout << "Blue";  break;
  default:    cout << "Error: Invalid color";
}
```

Enumeration Types (Cont'd)

- **NO** value can belong to more than one type

```
enum Weekday {Mon, Tue, Wed, Thu, Fri};
enum Weekend {Fri, Sat, Sun}; //Illegal
```

- Characters as Enumerator values

  The value of the identifier is the value of the character's numeric code.

  example:

  enum special_chars { backspace = '\b', bell = '\a', newline = '\n',
  return = '\r', tab = '\t', vtab = '\v' };

- same scope rules apply

  - o once a type has been defined, its name and the names of its constants, are known in all subprogram unless they are locally redefined

- another common mistake

```
enum GRADE {'A', 'B', 'C', 'D', 'F'};

What's wrong?
```