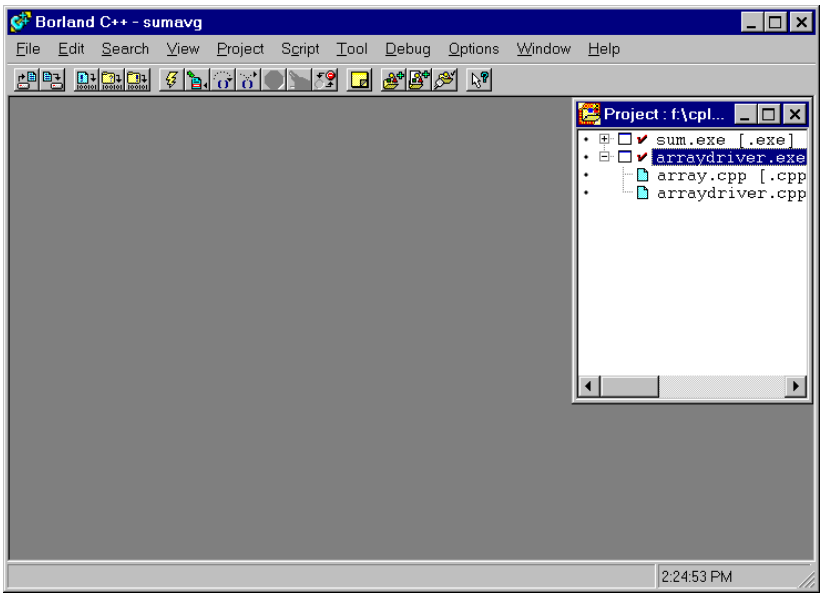
Separate Compilation

We can create a project composed of several C++ files. All the files belonging to one project should reside in the same directory on your disk. The best practice is to create a new directory for each new project.

Example1 : a program to process an array of numbers, using a separately compiled file. We have the following:

arraydriver.cpp	This file contains the "main" function definition. Execution will begin here.
array.h	Contains prototypes needed for the array.cpp file.
array.cpp	Contains the function definitions for the functions needed by arraydriver.cpp to process a data array.
arraydriver.exe	The is the *target* executable file created by the compiler. All the other files are subordinate to it.

Use #include "array.h" The " " indicate to the compiler to look for the file in the same directory as the program. < > tells the compiler to look in a specified *include* directory.



In Borland, right click on the target (.exe) node to add a dependent node (e.g., array.cpp). The .h header file does not have to be listed as a dependent node in the project window (only source files get listed there), but it does have to be in the same directory as the target.

Advantages of separate compilation:

- Compile once
- When you make changes to one file, you don't need to recompile all of them.
- Once a separately compiled function has been thoroughly tested (and debugged) there is no further need to be concerned about how it works. Black Box.

```
//array.h
void readarray (float[], int&);
void printarray (const float[], int);
float sumarray (const float[], int);
```

```
//array.cpp
#include <iostream>

void readarray(float x[], int &n){
   n = 0;   // n is the "number of numbers" in the array
   char more='y';
   while (more != 'n' && more != 'N'){
                        cout << "\nEnter a number:  "; cin >> x[n];
      n++;
      cout << "Another number? (y or n)  ";
      cin >> more;
      }
}
//*********************************************************************
float sumarray (const float x[], int n){
   float sum = 0;
   for (int i=0; i<n; i++) sum += x[i];
   return sum;
}
//*********************************************************************
   void printarray (const float x[], int n){
   cout << "\nThe data values are:\n";
   for (int i=0; i<n; i++) cout << '\t' << x[i];
   cout << endl;
}
```

```
//arraydriver.cpp

#include "array.h"
#include <iostream>

int main(){
   int n;
   float data [20] = {0.0};
   readarray(data, n);
   printarray(data, n);
   cout << "\nThe sum is  " << sumarray(data, n);
   cout << "\n\nPress any key to close window."; char c; cin >> c;
   return 0;
}
```

Example 2: Grading Program: Revise the programming assignment "A Very Grading Problem" to include separately compiled files. You may have the following:

grading.cpp	This file contains the "main" function definition. Execution will begin here.

maxmin.h	Contains prototypes needed for the maxmin.cpp file.
maxmin.cpp	Contains the function definitions for the max and min functions needed for grading.cpp
grading.exe	The is the ***target*** executable file created by the compiler.

Note that the functions for printing the report should really be in their own file. We do that when we create a printReport class.

```
//maxmin.h
//function prototypes for maxmin.cpp file

float FindMax(float);
float FindMin(float);
```

```
//maxmin.cpp
//function definitions

#include "maxmin.h"

float FindMax(float FinalGrade){
   static float MaxSoFar = -1;
   if (FinalGrade > MaxSoFar)
                        MaxSoFar = FinalGrade;
   return MaxSoFar;
}

float FindMin(float FinalGrade){
   static float MinSoFar = 150;
   if (FinalGrade < MinSoFar)
                        MinSoFar = FinalGrade;
   return MinSoFar;
}
```

```
// grading.cpp
//Programming Assignment - A Very Grading Problem
//using functions, arrays, I/O files, header file
#include "maxmin.h"
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
using namespace std;
void PrintSummaryInfo(int, float, float, float); //function prototypes
void PrintReportHeadings(string, string, float[4]);
void PrintColumnHeadings();
void PrintStudentLine(string, string, string, int[4], float);
void PrintaLine(int, char);
void SkipLines (int);
void indent (int);
ifstream infile ("f:\\cplus\\datafiles\\1students.dat");
ofstream outReport ("f:\\cplus\\reports\\grading.txt");
int main (){
   string id, course, semester, LastName, FirstName;
   float weight[4];
```

```
   int n = 0, grade[4];
   float FinalGrade, SumGrades = 0, HiGrade, LoGrade;
   if (!infile) cerr << "Error: could not open input file\n";
   infile >> course >> semester;
   for (int i=0; i<4; i++) infile >> weight[i];
   PrintReportHeadings(course, semester, weight);
   while (infile >> id >> LastName >> FirstName) {
      FinalGrade = 0;
      for (int i=0; i<4; i++){
         infile >> grade[i];
         FinalGrade +=  weight[i]*grade[i];
      }
      PrintStudentLine(id, LastName, FirstName, grade, FinalGrade);
      SumGrades += FinalGrade;
      n++;
      HiGrade = FindMax(FinalGrade);
      LoGrade = FindMin(FinalGrade);
   } //end while
   PrintSummaryInfo(n, SumGrades, HiGrade, LoGrade);
   cout << "Close console window?  "; char c; cin>>c;
   return 0;
} //end main
/*****************************************************************/
void PrintStudentLine(string id, string lName, string fName, int g[4],
float FinalGrade){
   outReport << setiosflags (ios::fixed) << setprecision(2)
      << setw (13) << setiosflags (ios::left) << setw (12)<< id
      << setw (10)<<fName << setw (10)<< lName
      << setiosflags(ios::right);
      for (int i=0; i<4; i++) outReport << setw (10) << g[i];
      outReport << setw (10) << FinalGrade << endl;
}
/*****************************************************************/
void PrintReportHeadings(string course, string semester, float w[4]){
   PrintaLine (80, ':');
   indent(20);
   outReport << "Little School of Soft Knocks\n";
   indent (24);
   outReport << "Student Grade Report" << endl;
   PrintaLine (80, ':');
   SkipLines (2);
   outReport << "\nSemester:  " << semester << "\nCourse:  " << course
      <<endl;
   SkipLines (2);
   PrintaLine (80, '.');
   SkipLines (2);
   outReport << "\nIn computing the Final Grade for each student, the
      following weights were used:\n";
   for (int i=0; i<4; i++) outReport << setw(10) << "Grade " << (i+1)
      << setw(10) << w[i] << endl;
   PrintaLine (80, '.');
   SkipLines (2);
   PrintColumnHeadings();
   PrintaLine (80, '.');
   outReport << endl;
}
/*****************************************************************/
```

```
void PrintColumnHeadings(){
   outReport << setiosflags(ios::left) << setw (13) << "Student ID" <<
setw (20) << "Name"<< setiosflags(ios::right)
      << setw (10) << "Grade 1" << setw (10)<< "Grade 2" << setw (10)<<
"Grade 3"
      << setw (10)<< "Grade 4" << setw (13) << "Final Grade" << endl;
}
/*****************************************************************/
void PrintSummaryInfo(int n, float sum, float max, float min){
   SkipLines (2);
   PrintaLine (80, ':');
   SkipLines (4);
   outReport << setiosflags(ios::showpoint | ios::fixed)
   << setw(40) << "number of students = " << n << endl
   << setw(40) << "Class Average = " << sum/n << endl
   << setw(40) << "Maximum Grade is " << max << endl
   << setw(40) << "Minimum Grade is " << min << endl;
   SkipLines (2);
PrintaLine (80, ':');
}
/*****************************************************************/
void PrintaLine(int n, char c){
for (int i=1; i<=n; i++)
   outReport << c;
outReport << endl;
}
/*****************************************************************/
void SkipLines (int n){
for (int i=1; i<=n; i++)
   outReport << endl;
}
/*****************************************************************/
void indent (int n){
for (int i=1; i<=n; i++)
   outReport << ' ';
}
```

Separate Compilation with Classes

Each class definition is placed in a header file - this is the specification, or user interface. The class's member functions are placed in a source code file (.cpp). The header files are included (#include) in each file in which the class is used. There is at lease once more source code (.cpp) file in which the program's "main" function is defined.

#ifndef TIME_H
#define TIME_H
…
#endif

This prevents multiple declarations in the program. In a large file several source files may require the same header file declarations. Sometimes one header file must "include" other header files. If the header file was already previously included the and TIME1_H was defined that the rest of the file (until #endif) is ignored.

Example:

```
//time.h from Deitel Fig 6.5 p. 379
//Declaration of the Time class
//member functions defined in time.cpp

//prevent multiple inclusions of header file
#ifndef TIME_H
#define TIME_H

class Time {
public:
   Time();
   void setTime (int, int, int);
   void printMilitary();
   void printStandard();
private:
   int hour;                        // 0 - 23
   int minute;                      // 0 - 59
   int second;                      // 0 - 59
};

#endif
```

```
//time.cpp
//member function definitions for Time class.
#include <iostream>
#include "time.h"

Time::Time() {hour = minute = second = 0;}
/*****************************************************************/
//sets time and tests for valid data
void Time::setTime (int h, int m, int s) {
   hour = (h >= 0 && h < 24) ? h : 0;
   minute = (m >= 0 && m < 60) ? m : 0;
   second = (s >= 0 && s < 60) ? s : 0;
}
/*****************************************************************/
void Time::printMilitary(){
   cout << (hour < 10 ? "0" : "") << hour << ":"
      << (minute < 10 ? "0" : "") << minute;
}
/*****************************************************************/
void Time::printStandard(){
   cout << ( (hour == 0 || hour == 12) ? 12: hour % 12)
      << ":" << (minute < 10 ? "0" : "") << minute
      << ":" << (second < 10 ? "0" : "") << second
      << (hour < 12 ? " AM" : " PM");
}
```

```
//timedriver.cpp
//from Deitel Fig. 6.5
```

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

int main(){
   Time t;
   cout << "The initial military time is ";
   t.printMilitary();
   cout << "\nThe initial standard time is ";
   t.printStandard();

   t.setTime(13, 27, 6);
   cout << "\n\nMilitary time after setTime is ";
   t.printMilitary();
   cout << "\nStandard time after setTime is ";
   t.printStandard();

   t.setTime(99, 99, 99);
   cout << "\n\nAfter attempting invalid settins:\n"
      << "Military time;  ";
   t.printMilitary();
   cout << "\nStandard time:  ";
   t.printStandard();
   cout << endl;
   cout << "\n\nPress any key to close window.  "; char c; cin >> c;
   return 0;
}
```