J. P. Cohoon and J. W. Davidson


# Functions

## Computational assistants

# Functions

- Previous examples
  - Programmer-defined functions
    - `main()`
    - `ApiMain()`
  - Library-defined functions
    - `cin.get()`
    - `string` member functions `size()`, `find()` and `substr()`
    - `RectangleShape` member function `Draw()`
    - `SimpleWindow` member function `Open()`
- Some good advice
  - Don’t reinvent the wheel! There are lots of libraries out there

# Terminology

- A function is invoked (called, executed) by a function call and produces a return value
- A function call specifies
  - The function name
    - Function name indicates what function is to be called
  - The values or *actual parameters* to be used in the invocation
    - The values are the actual information that the function requires to perform its task

```
#include <iostream>
#include <string>
using namespace std;
#include <math.h>
int main() {
   cout << "Quadratic coefficients: ";
   double a, b, c;
   cin >> a >> b >> c;
   if ((a != 0) && ((b*b - 4*a*c) > 0)) {
       double radical = sqrt(b*b - 4*a*c);
       double root1 = (-b + radical) / (2*a);
       double root2 = (-b - radical) / (2*a);
       cout << "Roots: " << root1 << " " << root2;
   }
   else {
       cout << "Does not have two real roots";
   }
   return 0;
}
```

Library header file

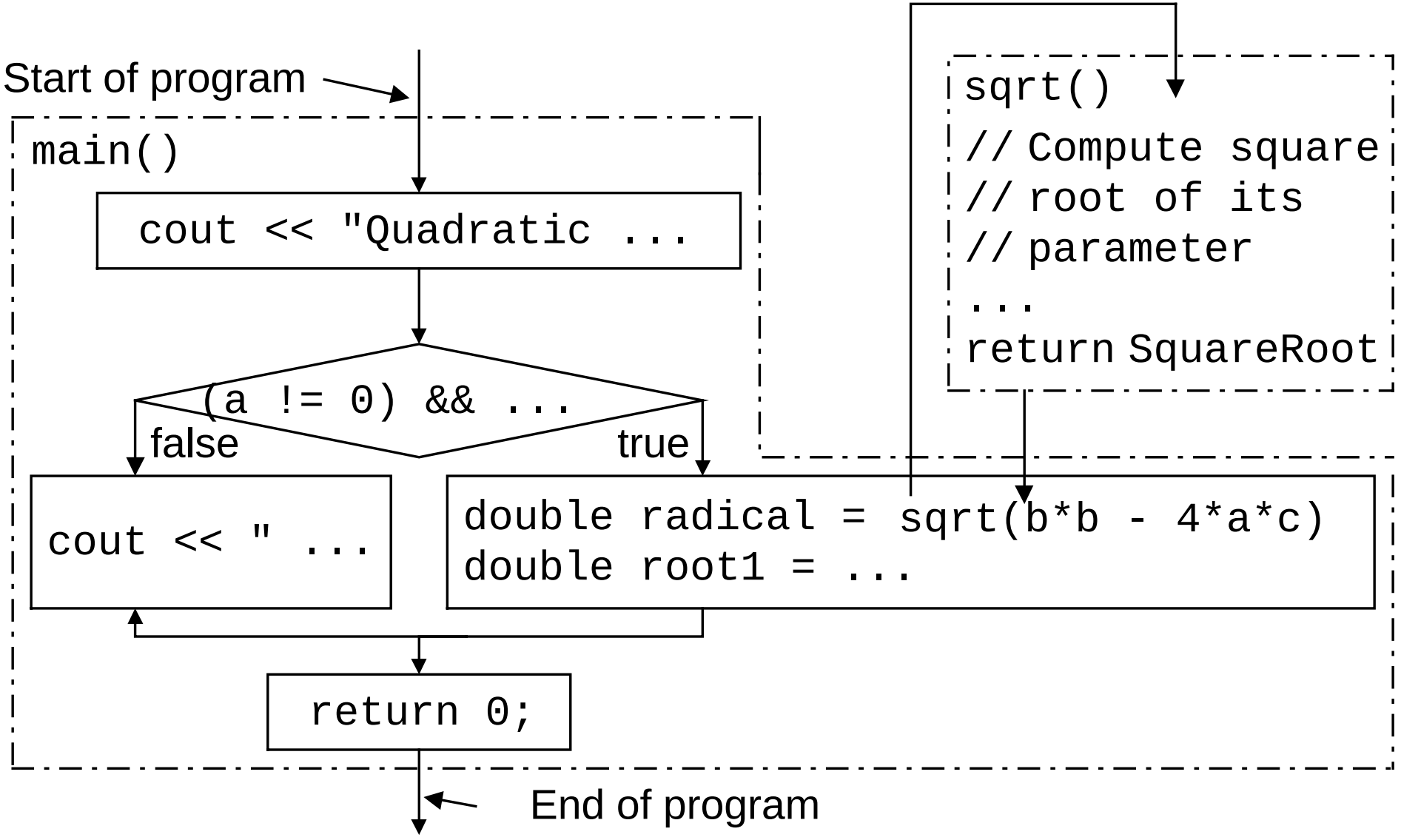
Invocation

# Invocation and Execution Process

- Flow of control is temporarily transferred to the invoked function
  - Next statement executed is the first one in the invoked function
- Correspondence established between the *actual* parameters used in the invocation with the *formal* parameters in the definition.
  - Value of `b*b - 4*a*c` $\rightarrow$ first actual parameter
  - Parameters are maintained in the invocation activation record
- Invoked function is executed
- Flow of control returns to the invocation statement
- The return value of the invoked function is used in the invocation expression

# Activation Records

- Fresh activation record is created for each invocation of a function.
- Even function `main()`, which is invoked by the operating system, has an activation record.
- The activation record for a function is large enough to store
  - Values associated with each object that is defined within the function
  - Memory to temporarily hold the return value
  - Depending upon the compiler
    - A pointer to the current statement being executed
    - A pointer to the invoking statement


Start of program
main()
cout << "Quadratic ...
(a != 0) && ...
false
true
cout << " ...
double radical = sqrt(b*b - 4*a*c)
double root1 = ...
return 0;
End of program
sqrt()
// Compute square
// root of its
// parameter
...
return SquareRoot

# Libraries

- Library
  - Collection of  functions, classes, and objects grouped by commonality of purpose
  - Include statement provides access to the names and descriptions of the library components
  - Linker connects program to actual library definitions
- Previous examples
  - String: STL’s string class
  - Graphics: EzWindows

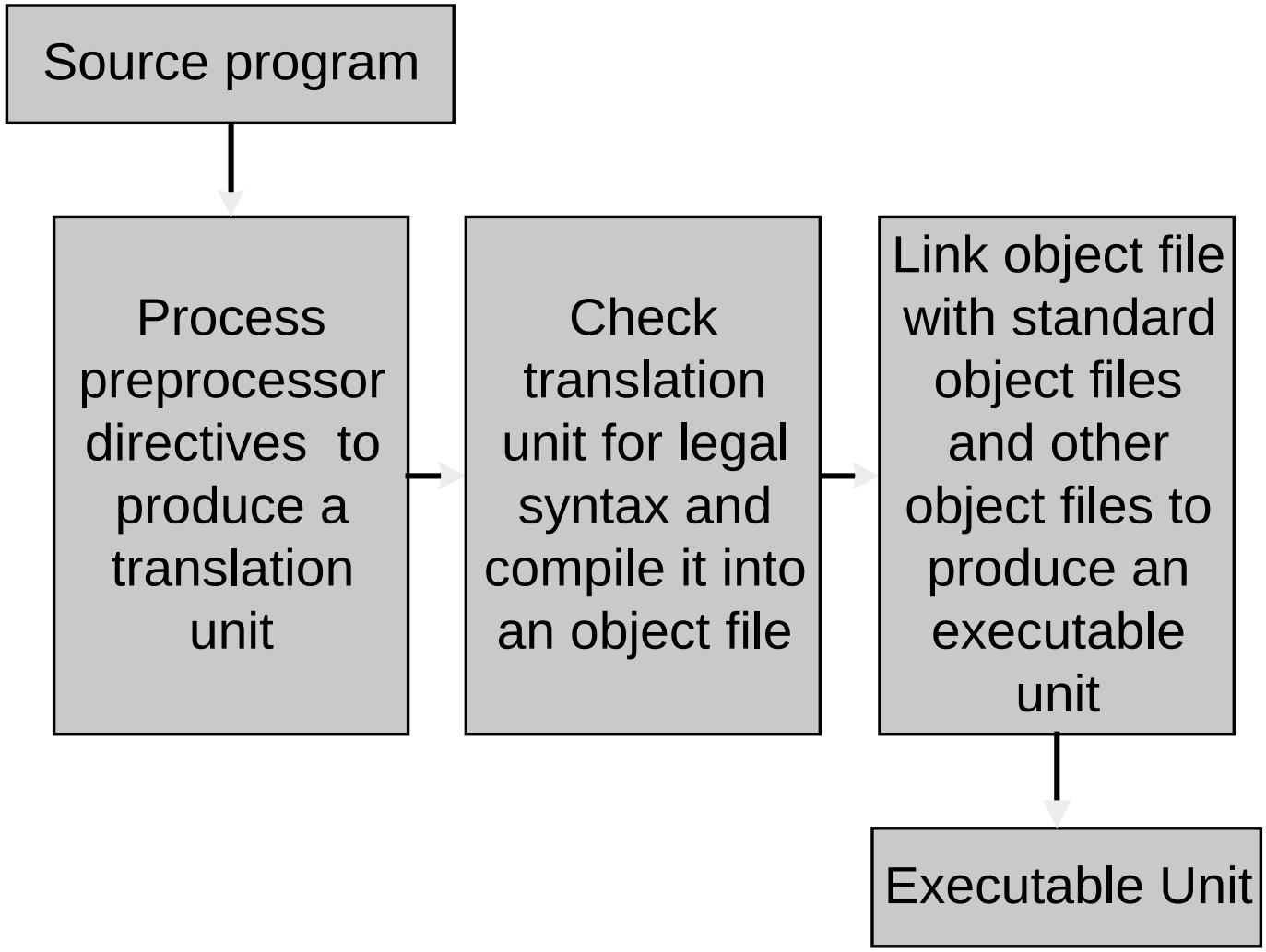
# Some Helpful Standard Libraries

- fstream
  - File stream processing
- assert
  - C-based library for assertion processing
- iomanip
  - Formatted input/output (I/O) requests
- ctype
  - C-based library for character manipulations
- math
  - C-based library for trigonometric and logarithmic functions
- Note
  - C++ has many other libraries

# Library Header Files

- Describes library components
- Typically contains
  - Function prototypes
    - Interface description
  - Class definitions
- Sometimes contains
  - Object definitions
    - Example: `cout` and `cin` in `iostream`
- Typically does not contain function definitions
  - Definitions are in source files
  - Access to compiled versions of source files provided by a linker

# Basic Translation Process

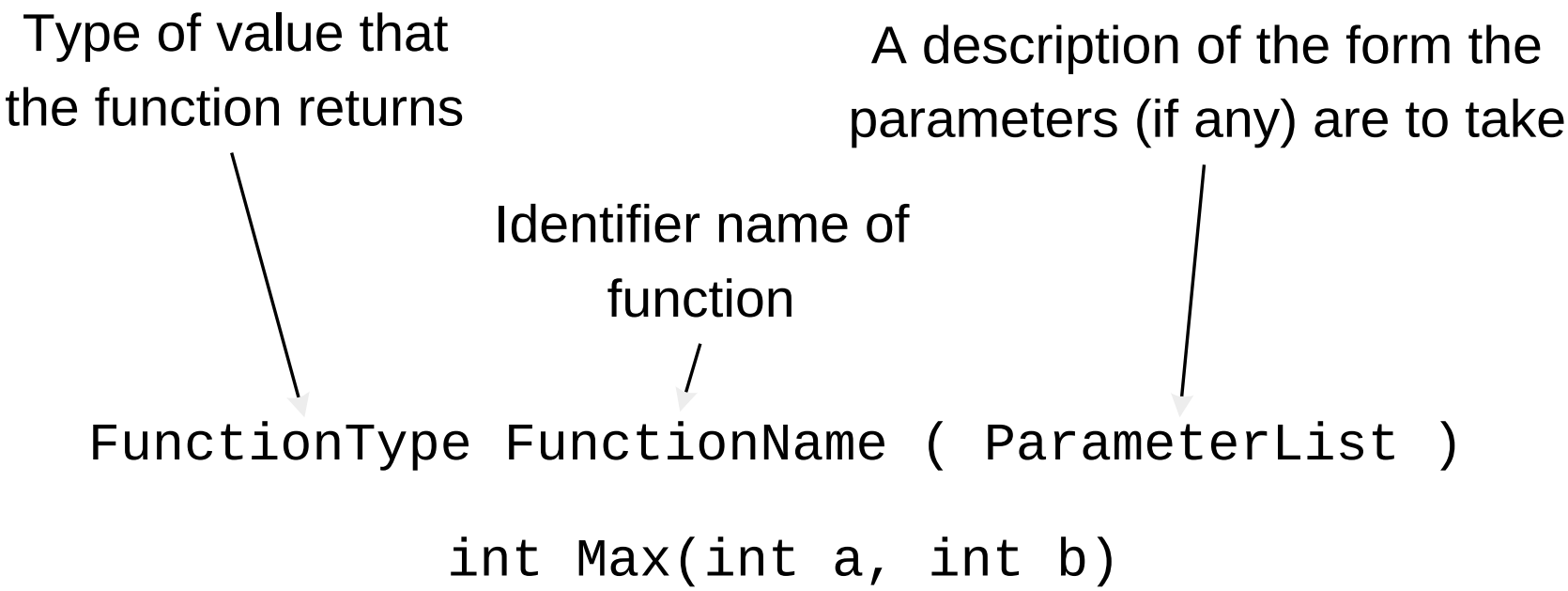


```
#include <fstream>     // file stream library
#include <string>
using namespace std;
int main() {
   ifstream fin("mydata.nbr");
   int ValuesProcessed = 0;
   float ValueSum = 0;
   float Value;
   while (fin >> Value) {
       ValueSum += Value;
       ++ValuesProcessed;
   }
   if (ValuesProcessed > 0) {
       ofstream fout("average.nbr");
       float Average = ValueSum / ValuesProcessed;
       fout << "Average: " << Average << endl;
   }
   else
       cerr << "No list to average" << endl;
   return 0;
}
```

```
ifstream sin("in1.txt");   // extract from in1.txt
ofstream sout("out1.txt"); // insert to out1.txt

string s;
while (sin >> s) {
  sout << s << endl;
}
sin.close();             // done with in1.txt
sout.close();            // done with out1.txt

sin.open("in2.txt");   // now extract from in2.txt
sout.open("out2.txt",  // now append to out2.txt
(ios_base::out | ios_base::app));

while (sin >> s) {
  sout << s << endl;
}

sin.close();// done with in2.txt
sout.close();// done with out2.txt
```

# Function Prototypes

- Before the function can appear in an invocation its interface must be specified
  - Prototype or complete definition



# Parameters

- Associations
  - The actual parameters are associated with the formal parameters based on position
    - First actual parameter is bound to first formal parameter
    - Second actual parameter is bound to second formal parameter
    - And so on
  - The names of the actual and formal parameters are immaterial in this mapping

```
#include <iostream>
#include <string>
using namespace std;
#include <math.h>
int main() {
   cout << "Principal amount: ";
   double StartingAmount;
   cin >> StartingAmount;
   cout << "Interest rate (%): ";
   double InterestRate;
   cin >> InterestRate;
   cout << "Years of deposit: ";
   double Years;
   cin >> Years;
```

```
    double EndingAmount = StartingAmount
     * pow(1+InterestRate/100.0, Years);
    cout << "Principal " << StartingAmount
     << " when compounded annually for " << Years
     << "\nyears at " << InterestRate
     << "% produces " << EndingAmount << endl;
    return 0;
}
```

```
#include <assert.h>
#include <iostream>
#include <string>
using namespace std;
int main() {
   int Numerator;
   cout << "Enter numerator: ";
   cin >> Numerator;
   int Denominator;
   cout << "Enter denominator: ";
   cin >> Denominator;
   assert(Denominator); // really should be if test
   int Ratio = Numerator / Denominator;
   int Remainder = Numerator % Denominator;
   cout << Numerator << "/" << Denominator << " = "
   << Ratio << " with remainder " << Remainder;
   return 0;
}
```