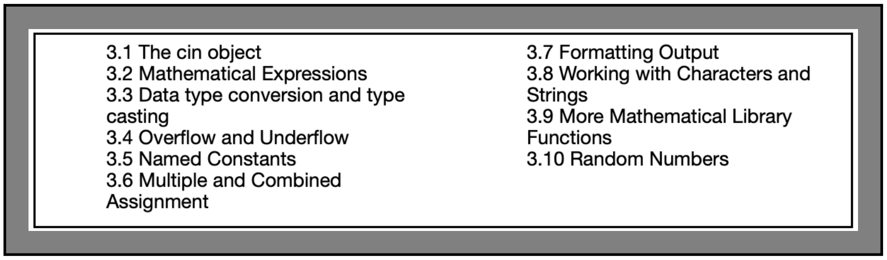
Chapter 3 - Expressions and Interactivity



3.1 The cin object
3.2 Mathematical Expressions
3.3 Data type conversion and type casting
3.4 Overflow and Underflow
3.5 Named Constants
3.6 Multiple and Combined Assignment
3.7 Formatting Output
3.8 Working with Characters and Strings
3.9 More Mathematical Library Functions
3.10 Random Numbers

Figure 1: Chapter outline

The big idea behind this chapter is ....

It relates to the previous chapter how ...

The main purpose of this chapter is ...

The key questions are ...

Why:

When:

How:

Why is this material at this point in the class?

You'll know this material when ...

Main assumptions are ...

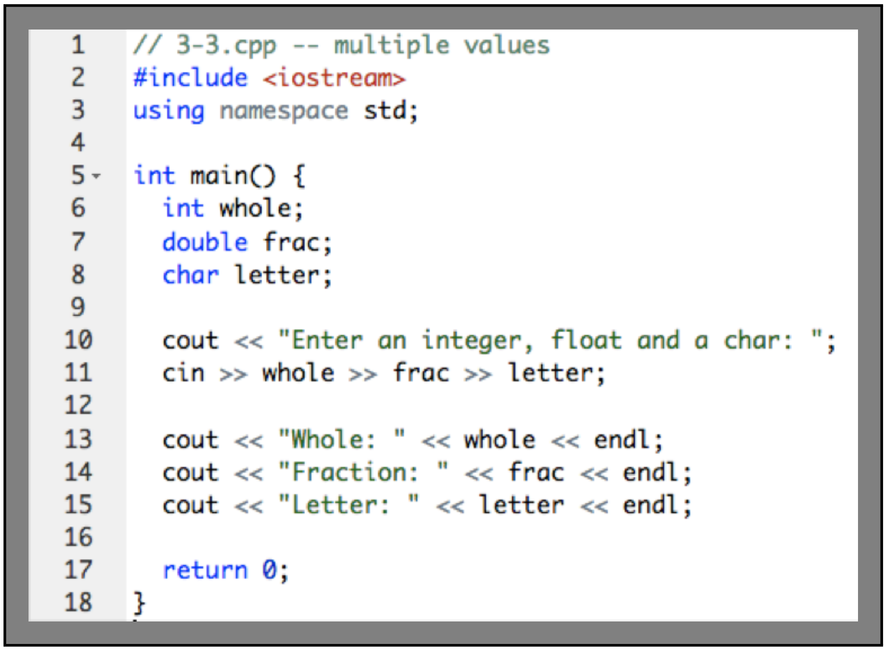
Opening Thoughts. Write any thoughts or questions you have before reading this material. See if you can find the answers while you read.

Key Ideas. Record major points from the chapter.

Idea	Notes

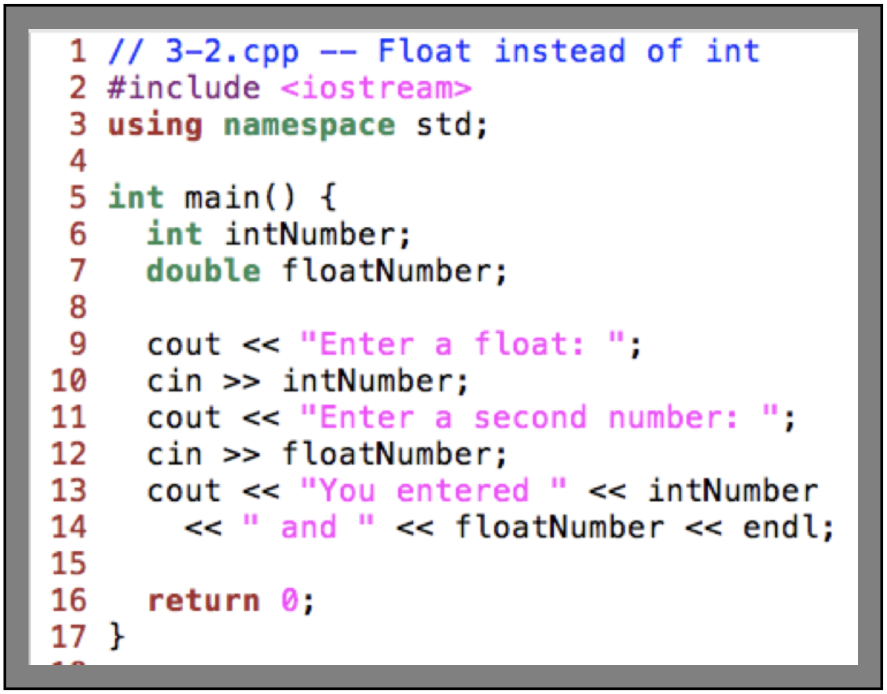
List of Figures

Demonstration Code.



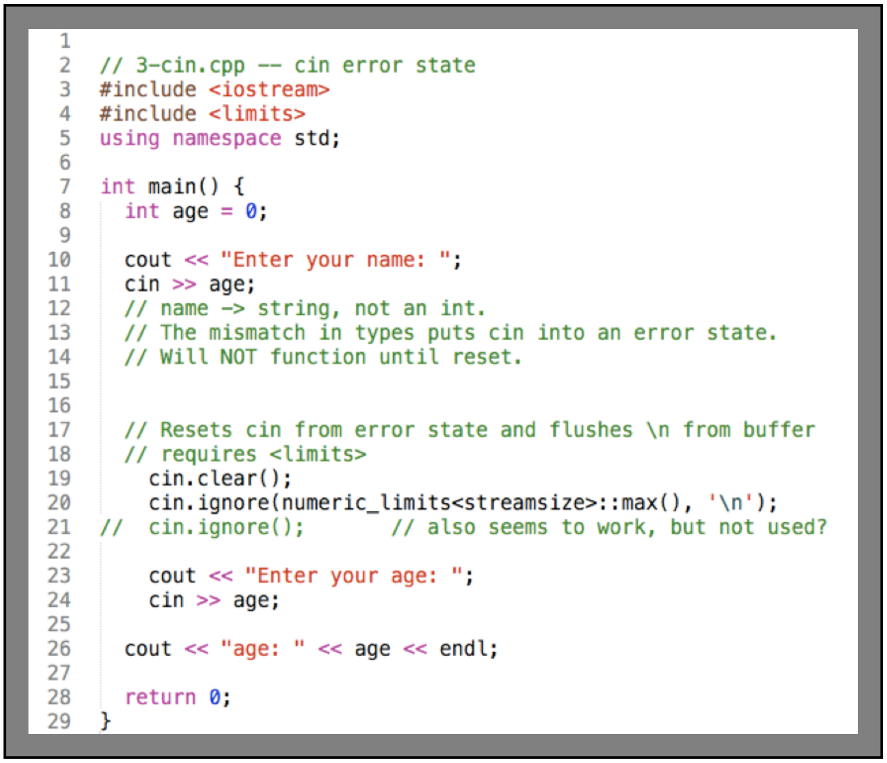
```cpp
// 3-3.cpp -- multiple values
#include <iostream>
using namespace std;

int main() {
  int whole;
  double frac;
  char letter;

  cout << "Enter an integer, float and a char: ";
  cin >> whole >> frac >> letter;

  cout << "Whole: " << whole << endl;
  cout << "Fraction: " << frac << endl;
  cout << "Letter: " << letter << endl;

  return 0;
}
```

Figure 2: Reading multiple values with 1 cin statement. Source file: 3-3.cpp



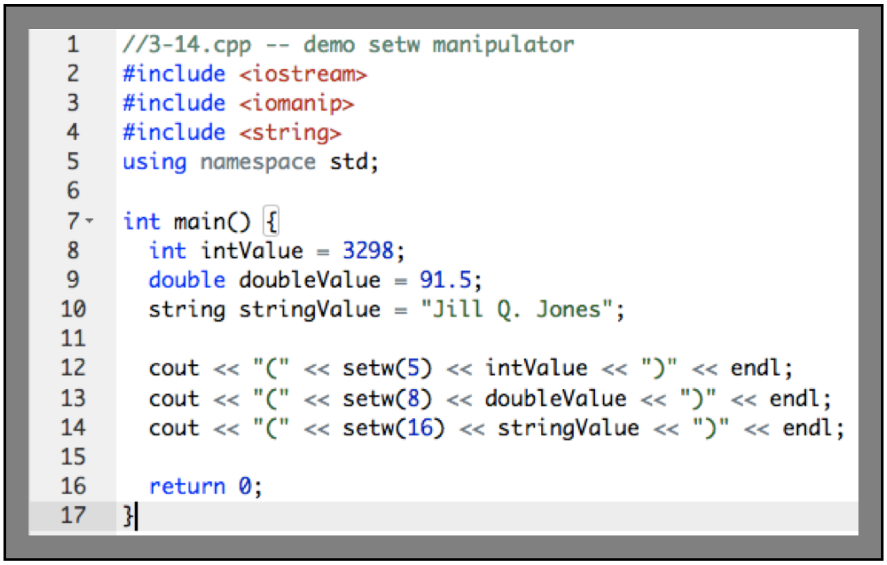
```cpp
// 3-2.cpp -- Float instead of int
#include <iostream>
using namespace std;

int main() {
  int intNumber;
  double floatNumber;

  cout << "Enter a float: ";
  cin >> intNumber;
  cout << "Enter a second number: ";
  cin >> floatNumber;
  cout << "You entered " << intNumber
    << " and " << floatNumber << endl;

  return 0;
}
```

Figure 3: Problem with cin and mismatch with data and type. Source file: 3-2.cpp



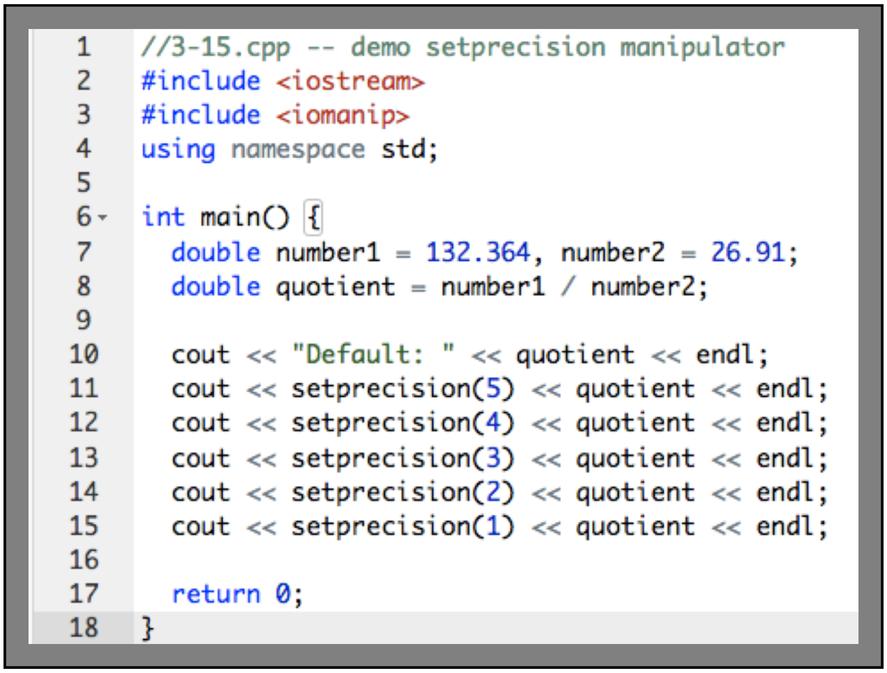
```cpp

// 3-cin.cpp -- cin error state
#include <iostream>
#include <limits>
using namespace std;

int main() {
  int age = 0;

  cout << "Enter your name: ";
  cin >> age;
  // name -> string, not an int.
  // The mismatch in types puts cin into an error state.
  // Will NOT function until reset.


  // Resets cin from error state and flushes \n from buffer
  // requires <limits>
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
//  cin.ignore();       // also seems to work, but not used?

    cout << "Enter your age: ";
    cin >> age;

  cout << "age: " << age << endl;

  return 0;
}
```

Figure 4: How to reset cin from error state. Source file: 3-cin.cpp



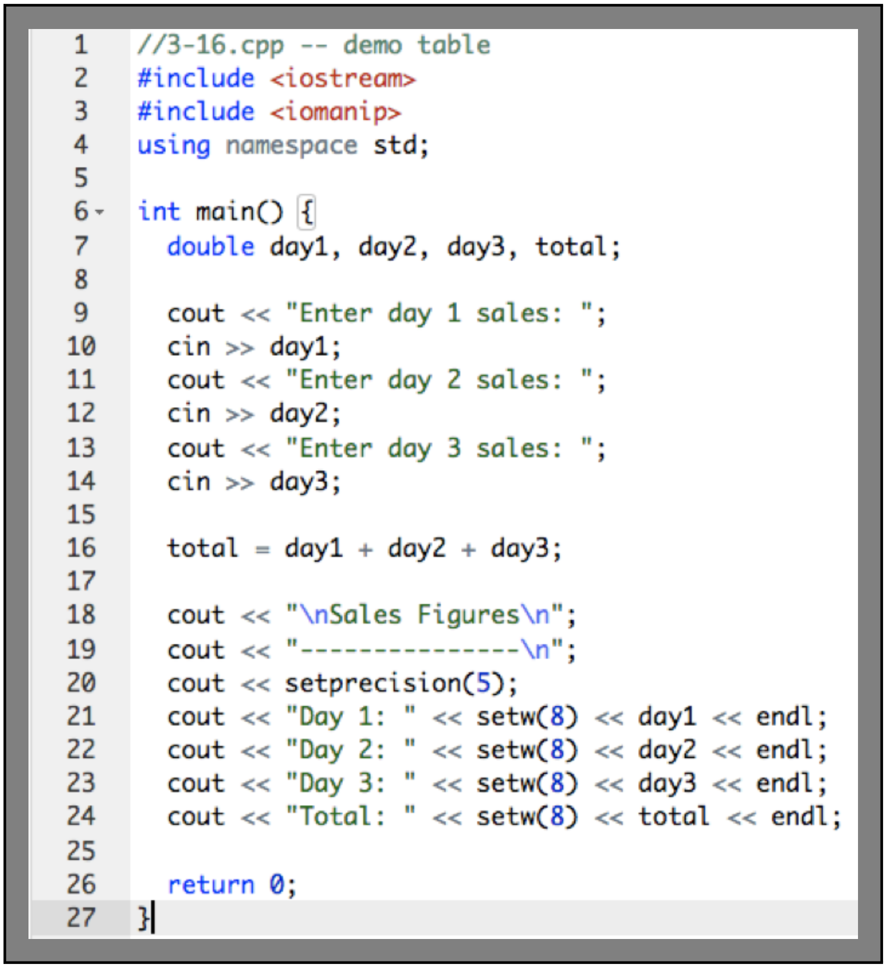
```cpp
//3-14.cpp -- demo setw manipulator
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

int main() {
  int intValue = 3298;
  double doubleValue = 91.5;
  string stringValue = "Jill Q. Jones";

  cout << "(" << setw(5) << intValue << ")" << endl;
  cout << "(" << setw(8) << doubleValue << ")" << endl;
  cout << "(" << setw(16) << stringValue << ")" << endl;

  return 0;
}
```

Figure 5: setw() - confine output to set filed length. Source file: 3-14.cpp



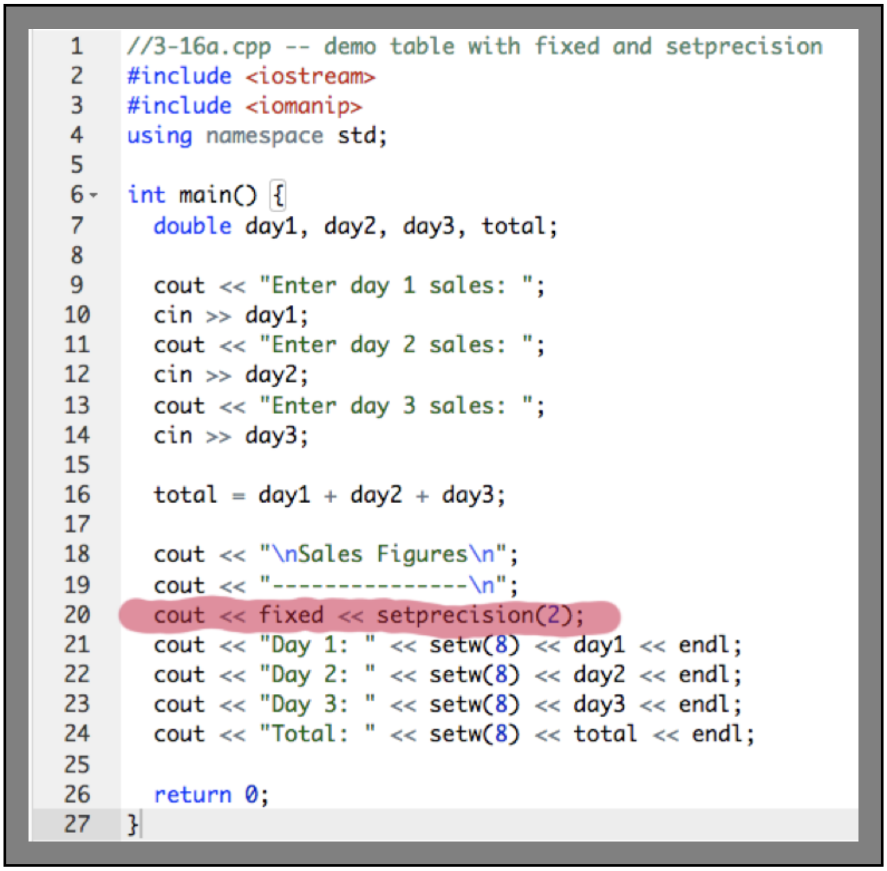
```cpp
//3-15.cpp -- demo setprecision manipulator
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
  double number1 = 132.364, number2 = 26.91;
  double quotient = number1 / number2;

  cout << "Default: " << quotient << endl;
  cout << setprecision(5) << quotient << endl;
  cout << setprecision(4) << quotient << endl;
  cout << setprecision(3) << quotient << endl;
  cout << setprecision(2) << quotient << endl;
  cout << setprecision(1) << quotient << endl;

  return 0;
}
```

Figure 6: setprecision() - sets # of significant digits for floats. Source file: 3-15.cpp



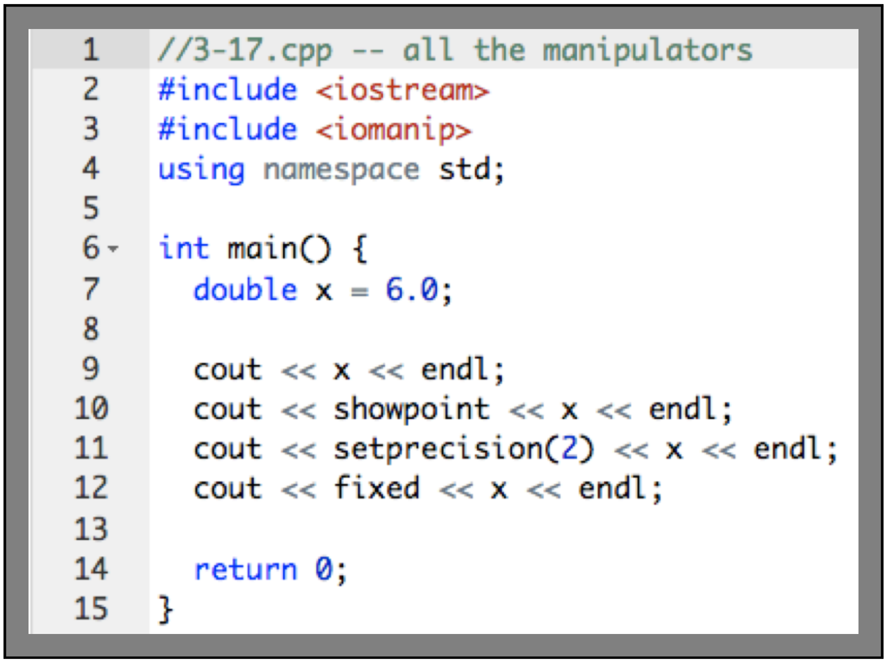
```cpp
//3-16.cpp -- demo table
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
  double day1, day2, day3, total;

  cout << "Enter day 1 sales: ";
  cin >> day1;
  cout << "Enter day 2 sales: ";
  cin >> day2;
  cout << "Enter day 3 sales: ";
  cin >> day3;

  total = day1 + day2 + day3;

  cout << "\nSales Figures\n";
  cout << "---------------\n";
  cout << setprecision(5);
  cout << "Day 1: " << setw(8) << day1 << endl;
  cout << "Day 2: " << setw(8) << day2 << endl;
  cout << "Day 3: " << setw(8) << day3 << endl;
  cout << "Total: " << setw(8) << total << endl;

  return 0;
}
```

Figure 7: Use setw and setprecision to build a table. Source file: 3-16.cpp



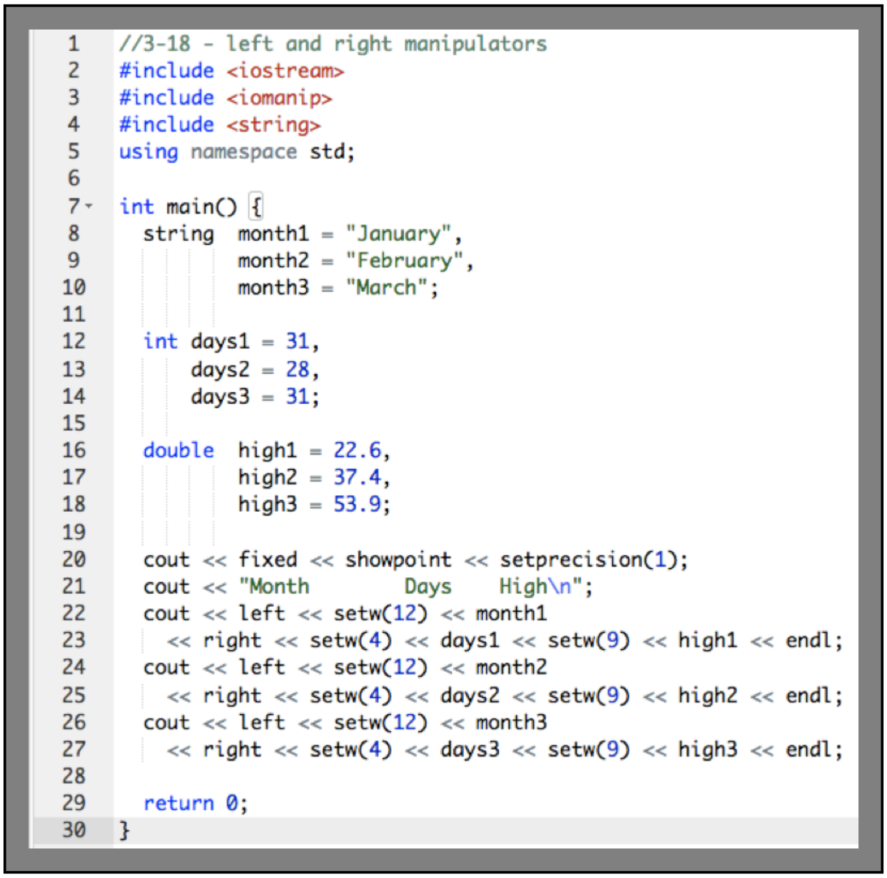
```cpp
//3-16a.cpp -- demo table with fixed and setprecision
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
  double day1, day2, day3, total;

  cout << "Enter day 1 sales: ";
  cin >> day1;
  cout << "Enter day 2 sales: ";
  cin >> day2;
  cout << "Enter day 3 sales: ";
  cin >> day3;

  total = day1 + day2 + day3;

  cout << "\nSales Figures\n";
  cout << "---------------\n";
  cout << fixed << setprecision(2);
  cout << "Day 1: " << setw(8) << day1 << endl;
  cout << "Day 2: " << setw(8) << day2 << endl;
  cout << "Day 3: " << setw(8) << day3 << endl;
  cout << "Total: " << setw(8) << total << endl;

  return 0;
}
```

Figure 8: fixed - suppress scientific notation. Source file: 3-16a.cpp



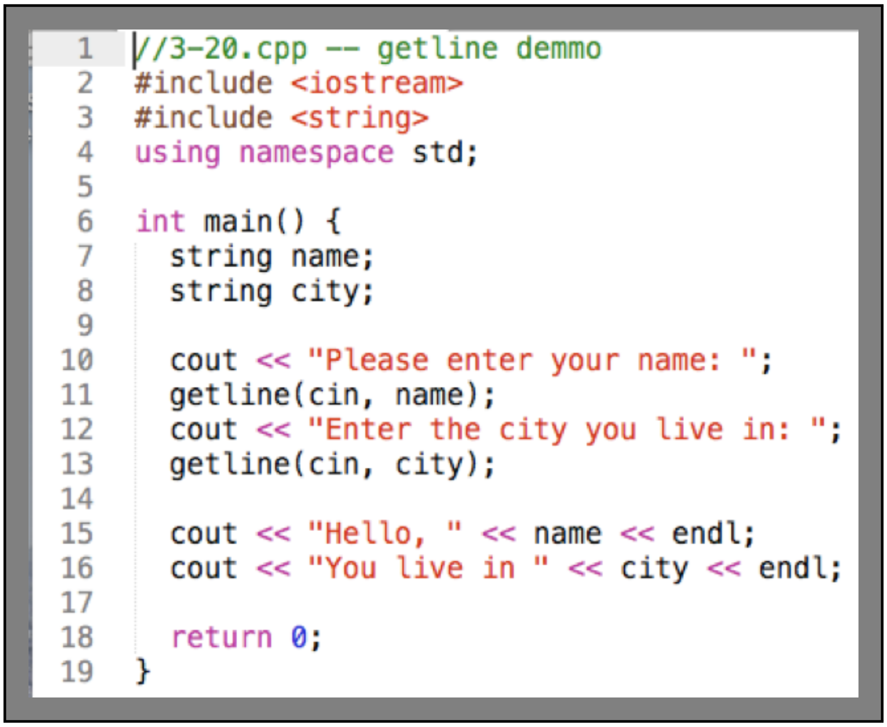
```cpp
//3-17.cpp -- all the manipulators
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
  double x = 6.0;

  cout << x << endl;
  cout << showpoint << x << endl;
  cout << setprecision(2) << x << endl;
  cout << fixed << x << endl;

  return 0;
}
```

Figure 9: showpoint() - show all digits, even zeros. Source file: 3-17.cpp



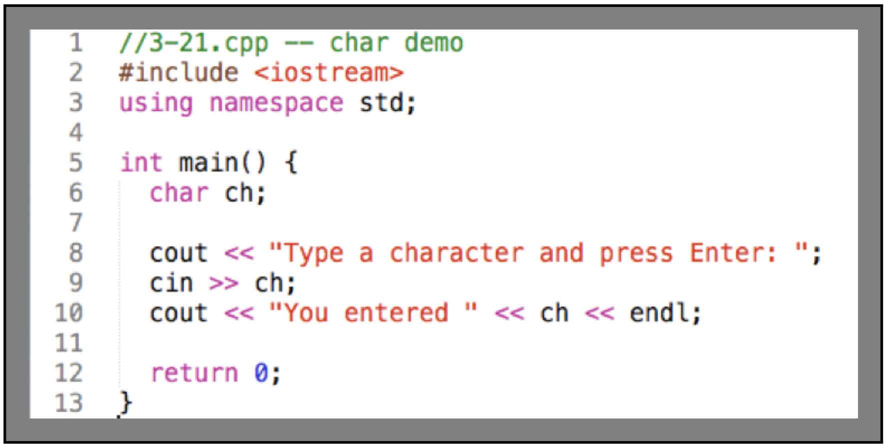
```
//3-18 - left and right manipulators
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

int main() {
  string  month1 = "January",
          month2 = "February",
          month3 = "March";

  int days1 = 31,
      days2 = 28,
      days3 = 31;

  double  high1 = 22.6,
          high2 = 37.4,
          high3 = 53.9;

  cout << fixed << showpoint << setprecision(1);
  cout << "Month        Days    High\n";
  cout << left << setw(12) << month1
    << right << setw(4) << days1 << setw(9) << high1 << endl;
  cout << left << setw(12) << month2
    << right << setw(4) << days2 << setw(9) << high2 << endl;
  cout << left << setw(12) << month3
    << right << setw(4) << days3 << setw(9) << high3 << endl;

  return 0;
}
```

Figure 10: left() and right() - control output alignment. Source file: 3-18.cpp



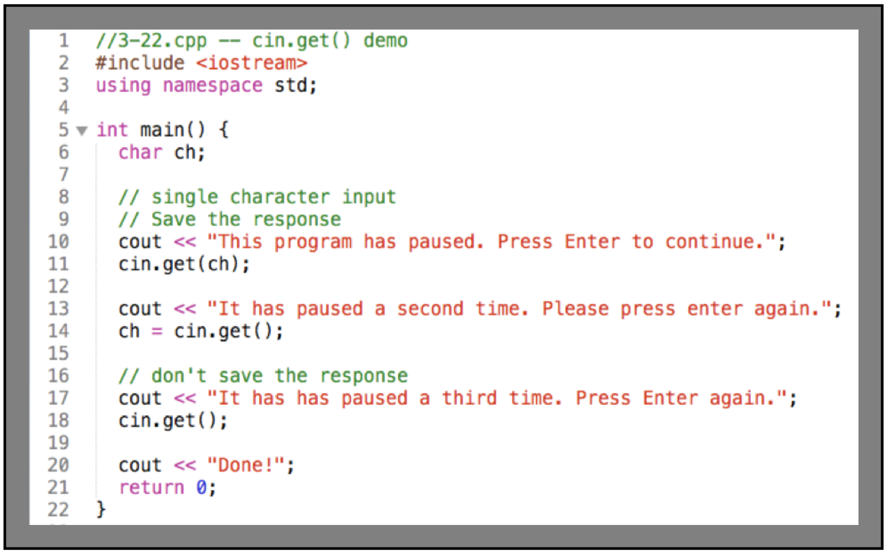
```
//3-20.cpp -- getline demmo
#include <iostream>
#include <string>
using namespace std;

int main() {
  string name;
  string city;

  cout << "Please enter your name: ";
  getline(cin, name);
  cout << "Enter the city you live in: ";
  getline(cin, city);

  cout << "Hello, " << name << endl;
  cout << "You live in " << city << endl;

  return 0;
}
```

Figure 11: §3.8 Demo of getline(). Source file: 3-20.cpp



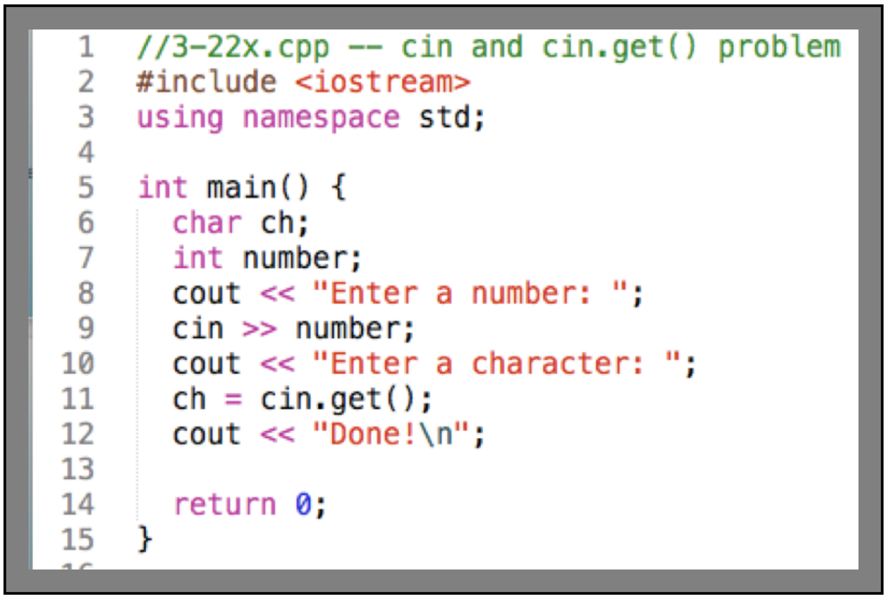
```cpp
//3-21.cpp -- char demo
#include <iostream>
using namespace std;

int main() {
  char ch;

  cout << "Type a character and press Enter: ";
  cin >> ch;
  cout << "You entered " << ch << endl;

  return 0;
}
```

Figure 12: §3.8 char demo. Source file: 3-21.cpp



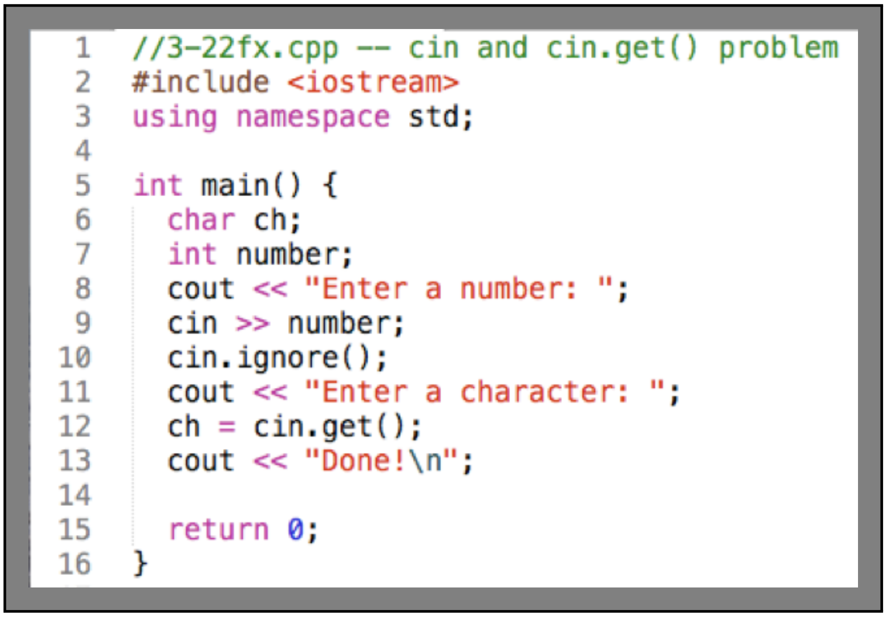
```cpp
//3-22.cpp -- cin.get() demo
#include <iostream>
using namespace std;

int main() {
  char ch;

  // single character input
  // Save the response
  cout << "This program has paused. Press Enter to continue.";
  cin.get(ch);

  cout << "It has paused a second time. Please press enter again.";
  ch = cin.get();

  // don't save the response
  cout << "It has has paused a third time. Press Enter again.";
  cin.get();

  cout << "Done!";
  return 0;
}
```

Figure 13: §3.8 Demo of cin.get(). Source file: 3-22.cpp



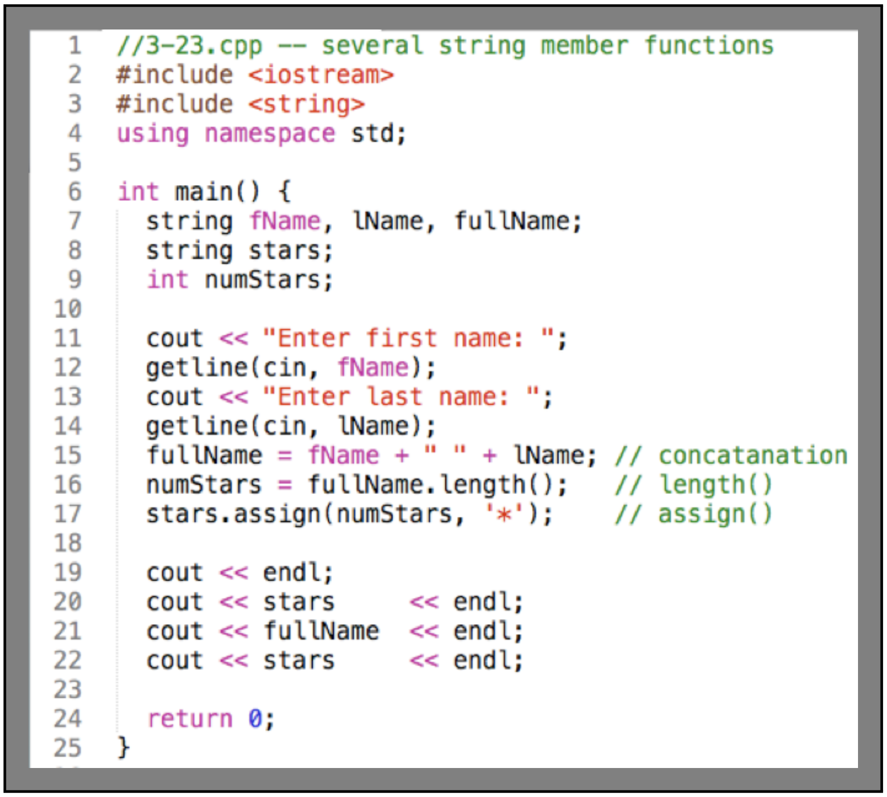
```cpp
//3-22x.cpp -- cin and cin.get() problem
#include <iostream>
using namespace std;

int main() {
  char ch;
  int number;
  cout << "Enter a number: ";
  cin >> number;
  cout << "Enter a character: ";
  ch = cin.get();
  cout << "Done!\n";

  return 0;
}
```

Figure 14: §3.8 Demo of getline(). Source file: 3-22x.cpp



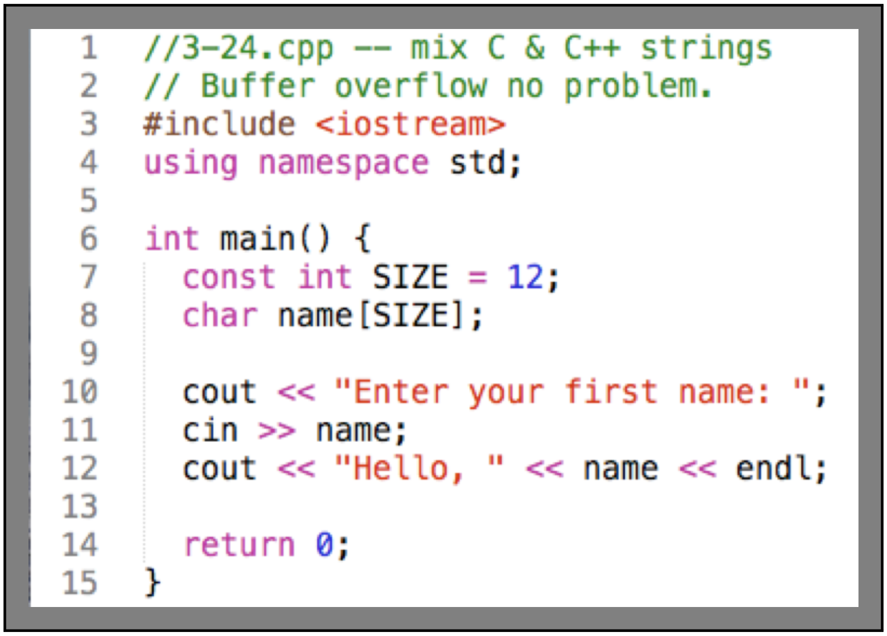
```cpp
//3-22fx.cpp -- cin and cin.get() problem
#include <iostream>
using namespace std;

int main() {
  char ch;
  int number;
  cout << "Enter a number: ";
  cin >> number;
  cin.ignore();
  cout << "Enter a character: ";
  ch = cin.get();
  cout << "Done!\n";

  return 0;
}
```

Figure 15: §3.8 Cin and cinget problem. Source file: 3-22fx.cpp



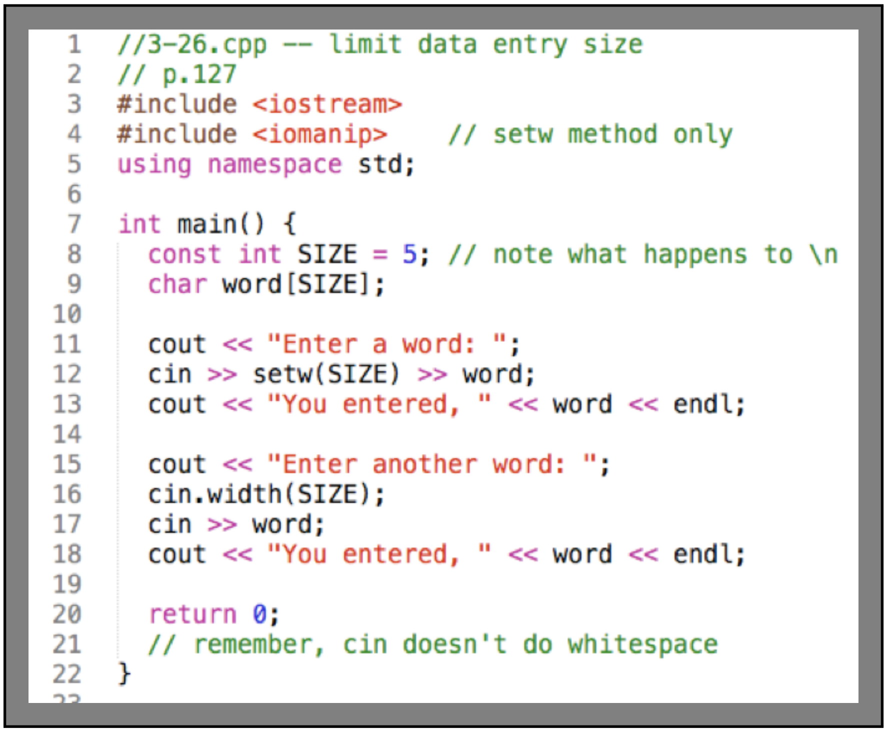
```cpp
//3-23.cpp -- several string member functions
#include <iostream>
#include <string>
using namespace std;

int main() {
  string fName, lName, fullName;
  string stars;
  int numStars;

  cout << "Enter first name: ";
  getline(cin, fName);
  cout << "Enter last name: ";
  getline(cin, lName);
  fullName = fName + " " + lName; // concatanation
  numStars = fullName.length();   // length()
  stars.assign(numStars, '*');    // assign()

  cout << endl;
  cout << stars     << endl;
  cout << fullName  << endl;
  cout << stars     << endl;

  return 0;
}
```

Figure 16: §3.8 String member function demo. Source file: 3-23.cpp



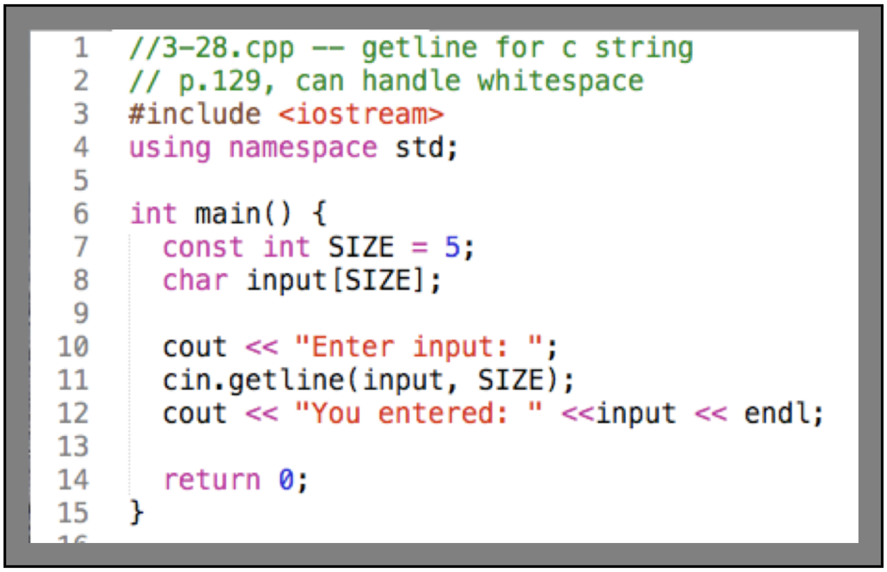
```
//3-24.cpp -- mix C & C++ strings
// Buffer overflow no problem.
#include <iostream>
using namespace std;

int main() {
  const int SIZE = 12;
  char name[SIZE];

  cout << "Enter your first name: ";
  cin >> name;
  cout << "Hello, " << name << endl;

  return 0;
}
```

Figure 17: §3.8 Demo of getline(). Source file: 3-24.cpp



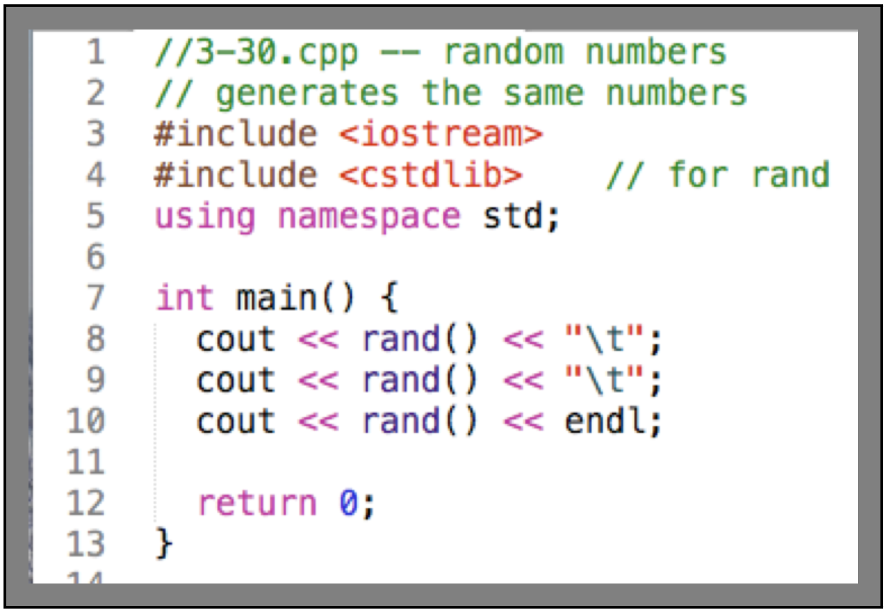
```
//3-26.cpp -- limit data entry size
// p.127
#include <iostream>
#include <iomanip>    // setw method only
using namespace std;

int main() {
  const int SIZE = 5; // note what happens to \n
  char word[SIZE];

  cout << "Enter a word: ";
  cin >> setw(SIZE) >> word;
  cout << "You entered, " << word << endl;

  cout << "Enter another word: ";
  cin.width(SIZE);
  cin >> word;
  cout << "You entered, " << word << endl;

  return 0;
  // remember, cin doesn't do whitespace
}
```

Figure 18: §3.8 How to limit the size of date entry. Source file: 3-26.cpp



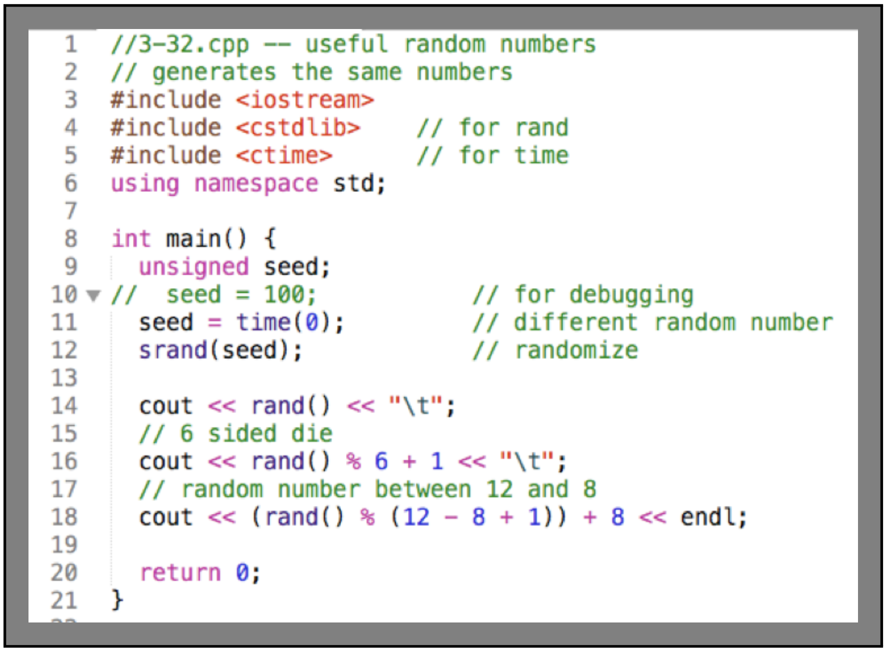
```cpp
//3-28.cpp -- getline for c string
// p.129, can handle whitespace
#include <iostream>
using namespace std;

int main() {
  const int SIZE = 5;
  char input[SIZE];

  cout << "Enter input: ";
  cin.getline(input, SIZE);
  cout << "You entered: " <<input << endl;

  return 0;
}
```

Figure 19: §3.8 Using getline() with c strings. Source file: 3-28.cpp



```cpp
//3-30.cpp -- random numbers
// generates the same numbers
#include <iostream>
#include <cstdlib>    // for rand
using namespace std;

int main() {
  cout << rand() << "\t";
  cout << rand() << "\t";
  cout << rand() << endl;

  return 0;
}
```

Figure 20: §3.10 Intro to random numbers. Source file: 3-30.cpp



```cpp
//3-32.cpp -- useful random numbers
// generates the same numbers
#include <iostream>
#include <cstdlib>    // for rand
#include <ctime>      // for time
using namespace std;

int main() {
  unsigned seed;
//  seed = 100;           // for debugging
  seed = time(0);         // different random number
  srand(seed);            // randomize

  cout << rand() << "\t";
  // 6 sided die
  cout << rand() % 6 + 1 << "\t";
  // random number between 12 and 8
  cout << (rand() % (12 - 8 + 1)) + 8 << endl;

  return 0;
}
```

Figure 21: §3.10 Using random numbers. Source file: 3-32.cpp