Tony Gaddis 5th Ed
Starting Out with C++

COMPUTER SCIENCE

CHAPTER 3
EXPRESSIONS AND INTERACTIVITY

Mohammad Al-Husseini

The cin Object

- So far all of your programs have had built in data
- Obviously, useful programs need to get input from outside
  - And store them in variables
- The cin object is the standard input object and accomplishes this task

The cin Object

```
1  // This program asks the user to enter the length and width of
2  // a rectangle. It calculates the rectangle's area and displays
3  // the value on the screen.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int length, width, area;
10
11    cout << "This program calculates the area of a ";
12    cout << "rectangle.\n";
13    cout << "What is the length of the rectangle? ";
14    cin >> length;
15    cout << "What is the width of the rectangle? ";
16    cin >> width;
17    area = length * width;
18    cout << "The area of the rectangle is " << area << ".\n";
19    return 0;
20 }
```

The cin Object

- Gathering input from a user is normally a two step process
  - Use the cout object to display a prompt
  - Use the cin object to read a value
- #include <iostream> is required for both the cout and cin objects
- cout << “Something”; // stream insertion op
- cin >> variable; // stream extraction operator
  - Think of these like arrows, pointing where the data is going to or coming from

The cin Object

- Note that the cout object completes and continues
- But the cin object waits for the user to press [Enter] before continuing the code

```
 9     int length, width, area;
10
11     cout << "This program calculates the area of a ";
12     cout << "rectangle.\n";
13     cout << "What is the length of the rectangle? ";
14     cin >> length;
15     cout << "What is the width of the rectangle? ";
16     cin >> width;
17     area = length * width;
18     cout << "The area of the rectangle is " << area << ".\n";
19     return 0;
```

cin Multiple Values

```
1  // This program asks the user to enter the length and width of
2  // a rectangle. It calculates the rectangle's area and displays
3  // the value on the screen.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int length, width, area;
10
11    cout << "This program calculates the area of a ";
12    cout << "rectangle.\n";
13    cout << "Enter the length and width of the rectangle ";
14    cout << "separated by a space.\n";
15    cin >> length >> width;
16    area = length * width;
17    cout << "The area of the rectangle is " << area << endl;
18    return 0;
19 }
```

- 10 20[Enter]

cin Multiple Values

```
1  // This program demonstrates how cin can read multiple values
2  // of different data types.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int whole;
9     double fractional;
10    char letter;
11
12    cout << "Enter an integer, a double, and a character: ";
13    cin >> whole >> fractional >> letter;
14    cout << "Whole: " << whole << endl;
15    cout << "Fractional: " << fractional << endl;
16    cout << "Letter: " << letter << endl;
17    return 0;
18 }
```

- 5.7 4 b[Enter]

The Keyboard Buffer

- Enter an integer, a double, and a character: 5.7 4 b[Enter]
- int = 5
- double = .7
- char = 4
- b is left in the keyboard buffer for the next cin
- Be careful when mixing multiple inputs

Reading Strings

- Cannot store a string (>2 characters) in a char
- Use a character array to store strings instead
  - char company[12];
  - Must be large enough to hold the number of characters in the string + 1 (for the \0)
  - So company can hold 11 characters, since the last character will be the null terminator
  - Be warned, if you store a larger string, it will accept it and run past its bounds to erase other data (overflow)

Reading Strings

```
1  // This program reads two strings into two character arrays.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     char first[16], last[16];
8
9     cout << "Enter your first and last names and I will\n";
10    cout << "reverse them.\n";
11    cin >> first >> last;
12    cout << last << ", " << first << endl;
13    return 0;
14 }
```

- Johnny Jones[Enter]
- Jones, Johnny

- You could do this as one string but cin doesn't accept white space (solution coming soon)

Mathematical Expressions

- sum = 21 + 3;
  - The value of 21 + 3 must be determined
  - The addition operator returns the value 24
    - sum = 24;
  - sum is then set equal to 24 by the assignment operator
  - And the expression is completed
- Don’t be tempted to store every calculation into a variable
  - If you only use it once, you don’t need to store it

Mathematical Expressions

```
1  // This program asks the user to enter the numerator
2  // and denominator of a fraction and it displays the
3  // decimal value.
4
5  #include <iostream>
6  using namespace std;
7
8  int main()
9  {
10    double numerator, denominator;
11
12    cout << "This program shows the decimal value of ";
13    cout << "a fraction.\n";
14    cout << "Enter the numerator: ";
15    cin >> numerator;
16    cout << "Enter the denominator: ";
17    cin >> denominator;
18    cout << "The decimal value is ";
19    cout << (numerator / denominator) << endl;
20    return 0;
21 }
```

Overflow and Underflow

- If you assign a value to a variable that is too large for it to hold, then it overflows
- If you assign a value to a variable that is too low (negative) for it to hold, then it underflows
- You will get unexpected results
- You will search for bugs, but there will be none
- Look for a wrap around behavior

Overflow and Underflow

```
1  // This program demonstrates integer overflow and underflow.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     // testVar is initialized with the maximum value for a short.
8     short testVar = 32767;
9
10    // Display testVar.
11    cout << testVar << endl;
12
13    // Add 1 to testVar to make it overflow.
14    testVar = testVar + 1;
15    cout << testVar << endl;
16
17    // Subtract 1 from testVar to make it underflow.
18    testVar = testVar - 1;
19    cout << testVar << endl;
20    return 0;
21 }
```

- 32767
- -32768
- 32767

Operator Precedence

- outcome = 12 + 10 / 5 * 2 - 8;
  - outcome = 12 + 2 * 2 - 8;
  - outcome = 12 + 4 - 8;
  - outcome = 16 - 8;
  - outcome = 8;
- Left to Right
- When two operands are shared, the operator with the highest presence activates first
- result = 1 + 1 + 1 * 5;
  - result = 2 + 1 * 5;

Operator Precedence

- Operators from highest to lowest
  - ()
  - - unary negation
  - * / %
  - + -
  - =

Operator Associativity

- The order that an operator works in
  - - unary negation
    - Right to left
  - * / %
    - Left to right
    - ex = 4 * 4 / 4;
      - ex = 16 / 4;
  - + -
    - Left to right

Multiple Assignment

- You may assign values to multiple variables at once
  - int a, b, c, d;
    a = b = c = d = 12;
  - Stores the value 12 in each variable
  - This works because the assignment operator works right to left and returns its value upon completion
    - a = b = c = (d = 12);
    - a = b = (c = 12);
    - a = (b = 12);
    - a = 12;
    - 12

Combined Assignment

- You can perform simple arithmetic with assignment in one statement
  - number = number + 2;
  - number += 2;
- +=
- -=
- *=
- /=
- %=

Combined Assignment

```
1  // This program tracks the inventory of three widget stores
2  // that opened at the same time. Each store started with the
3  // same number of widgets in inventory. By subtracting the
4  // number of widgets each store has sold from its inventory,
5  // the current inventory can be calculated.
6  #include <iostream>
7  using namespace std;
8
9  int main()
10 {
11    int begInv,     // Begining inventory for all stores
12        sold,       // Number of widgets sold
13        store1,     // Store 1's inventory
14        store2,     // Store 2's inventory
15        store3;     // Store 3's inventory
16
17    // Get the beginning inventory for all the stores.
18    cout << "One week ago, 3 new widget stores opened\n";
19    cout << "at the same time with the same beginning\n";
20    cout << "inventory. What was the beginning inventory?\n";
21    cin >> begInv;
22
23    // Set each store's inventory.
24    store1 = store2 = store3 = begInv;
```

Combined Assignment

```
26      // Get the number of widgets sold at store 1.
27      cout << "How many widgets has store 1 sold? ";
28      cin >> sold;
29      store1 -= sold; // Adjust store 1's inventory.
30
31      // Get the number of widgets sold at store 2.
32      cout << "How many widgets has store 2 sold? ";
33      cin >> sold;
34      store2 -= sold; // Adjust store 2's inventory.
35
36      // Get the number of widgets sold at store 3.
37      cout << "How many widgets has store 3 sold? ";
38      cin >> sold;
39      store3 -= sold; // Adjust store 3's inventory.
40
41      // Display each store's current inventory.
42      cout << "\nThe current inventory of each store:\n\n";
43      cout << "Store 1: " << store1 << endl;
44      cout << "Store 2: " << store2 << endl;
45      cout << "Store 3: " << store3 << endl;
46      return 0;
47  }
```

Converting Algebra

- $a = \frac{3x + 2}{4a - 1}$
- a = (3 * x + 2) / (4 * a - 1);
- Fairly straightforward for the operators you know
  - Note: there is no exponent operator in C++
  - You must use a function contained in the cmath library
- You may also express numbers in e notation
  - double num1 = 3.2e10;
  - double num2 = 1.8e-2;

Exponents

- #include <cmath>
- area = pow(4.0, 2.0);
  - Function's name is pow
  - Parenthesis contains the two arguments
  - (double base, double exponent)
  - Returns the result of 4^2 as a double
- area_of_circle = 3.14159 * pow(radius, 2.0);
  - You can use a function as part of an expression
  - And you can use variables as arguments to the function

More cmath Functions

Function	Example	Description
cos	y = cos(x);	Returns the cosine of the radian double x
sin	y = sin(x);	Returns the sine of the radian double x
tan	y = tan(x);	Returns the tangent of the radian double x
log	y = log(x);	Returns the natural log of the double x
log10	y = log10(x);	Returns the base-10 log of the double x
exp	y = exp(x);	Returns the base-e exponential function of the double x
abs	y = abs(x);	Returns the absolute value of the integer x
sqrt	y = sqrt(x);	Returns the square root of the double x
fmod	y = fmod(x, z);	Returns the remainder of x ; z as a double Same as modulus but for floating point Make certain that z is non-zero

- You can combine functions to make complex math
  - c = sqrt(pow(a, 2) + pow(b, 2));

Random Numbers

- Requires #include<cstdlib>
- rand()
  - Returns a random number
- Well… pseudorandom
  - The results will always be the same, every time you run the program
  - cout << rand() << endl;
    cout << rand() << endl;
    - Run 1: 12
      3
    - Run 2: 12
      3

Random Numbers

- Use srand(seed) to become more random

```
1  // This program demonstrates random numbers.
2  #include <iostream>
3  #include <cstdlib>   // rand and srand
4  #include <ctime>     // For the time function
5  using namespace std;
6
7  int main()
8  {
9     // Get the system time.
10    unsigned seed = time(0);
11
12    // Seed the random number generator.
13    srand(seed);
14
15    // Display three random numbers.
16    cout << rand() << endl;
17    cout << rand() << endl;
18    cout << rand() << endl;
19    return 0;
20 }
```

Random Numbers

- You can manipulate the range of random numbers with some clever math
  - y = 1 + rand() % maxValue;
    - y = 1 + rand() % 100; // Rand varies 1-100
    - Assume rand returns 37894
      37894 % 100 = 94
      94 + 1 = 95
    - Assume rand returns 500
      500 % 100 = 0
      0 + 1 = 1
    - Assume rand returns 2999
      2999 % 100 = 99
      99 + 1 = 100

Random Numbers

- We can take this one step further
  - y = rand() % range + minValue
    where range = maxValue - minValue + 1
    - Assume we want a number between 3 and 12
      range = 12 – 3 + 1 = 10
      y = rand() % 10 + 3
      - Assume rand returns 37894
        37894 % 10 = 4
        4 + 3 = 7
      - Assume rand returns 500
        500 % 10 = 0
        0 + 3 = 3
      - Assume rand returns 2999
        2999 % 10 = 9
        9 + 3 = 12

Type Conversion

- What happens when an int is multiplied by a float, what data type is the result?
  - C++ automatically promotes or demotes the operands to be the same type (type coercion)
- Data Type Ranking
  - long double
  - double
  - float
  - unsigned long
  - long (if int is the same size as long then demote 1)
  - unsigned int
  - int

Type Conversion

- Rules
  1. chars, shorts, and unsigned shorts are automatically promoted to int
  2. If an operator works on operands of different types, then the lower-ranking value is promoted to the higher-ranking one
     - int * double -> double * double
  3. When the final value of an expression is assigned to a variable, it is converted to the type of the variable
     - long double = int * double
       - long double = double
       - long double

Integer Division

- Whenever you divide an integer by an integer, the result will be an integer and any remainder will be discarded
  - double parts;
    parts = 21 / 10;
    - double = int / int;
    - double = int;
    - parts = 10;
    - Now the 10 gets promoted to a double to store in it in parts, but the remainder was already lost

Type Casting

- You are able to manually promote or demote
  - static_cast<DataType>(Value)
  - double number = 3.7;
    int val;
    val = static_cast<int>(number);
- Practical for avoiding integer division with variables

Type Casting

```
1  // This program uses a type cast to avoid integer division.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int books;          // Number of books to read
8     int months;         // Number of months spent reading
9     double perMonth;    // Average number of books per month
10
11    cout << "How many books do you plan to read? ";
12    cin >> books;
13    cout << "How many months will it take you to read them? ";
14    cin >> months;
15    perMonth = static_cast<double>(books) / months;
16    cout << "That is " << perMonth << " books per month.\n";
17    return 0;
18 }
```

- perMonth = static_cast<double>(books / months);
  Integer division happens before the cast (wrong)

Type Casting

```
1  // This program uses a type cast expression to print a character
2  // from a number.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int number = 65;
9
10    // Display the value of the number variable.
11    cout << number << endl;
12
13    // Display the value of number converted to
14    // the char data type.
15    cout << static_cast<char>(number) << endl;
16    return 0;
17 }
```

- 65
  A

Type Casting

- C++ also supports two older methods for type casting
  - perMonth = (double)books / months;
  - perMonth = double(books) / months;
- static_cast<>() is preferable however
  - Checked by the compiler
  - Clearer/More easily found

Named Constants

- Uses the const keyword before the data type
- Value cannot change after creation
- Convention suggests it be named with all capital letters and underscores
- Generally declared at the top of the code
- Good to use because if the value needs to change, you only have to change it in one place
  - Its also clearer to read as well

Named Constants

```
1  // This program calculates the area of a circle.
2  // The formula for the area of a circle is PI times
3  // the radius squared. PI is 3.14159.
4  #include <iostream>
5  #include <cmath>   // needed for pow function
6  using namespace std;
7
8  int main()
9  {
10    const double PI = 3.14159;
11    double area, radius;
12
13    cout << "This program calculates the area of a circle.\n";
14    cout << "What is the radius of the circle? ";
15    cin >> radius;
16    area = PI * pow(radius, 2.0);
17    cout << "The area is " << area << endl;
18    return 0;
19 }
```

Named Constants

```
1  // This program calculates the area of a circle.
2  // The formula for the area of a circle is PI times
3  // the radius squared. PI is 3.1459.
4  #include <iostream>
5  #include <cmath>   // needed for pow function
6  using namespace std;
7
8  #define PI 3.14159
9
10 int main()
11 {
12    double area, radius;
13
14    cout << "This program calculates the area of a circle.\n";
15    cout << "What is the radius of the circle? ";
16    cin >> radius;
17    area = PI * pow(radius, 2.0);
18    cout << "The area is " << area << endl;
19    return 0;
20 }
```

- More memory efficient (no variable created)

Formatting Output

- The following are all the same number:
  - 720
  - 720.0
  - 720.000000000
  - 720
  - 7.2e+2
  - +720
- You can control how these numbers are displayed and more

setw()

- Causes cout to display the data in a field of a certain size (width)
- The argument is an integer representing the width of the field
- It pads the field with leading spaces by default
  - You can later adjust its justification to be left and change what character it fills the field with
- If the field is not large enough for the data, it will be displayed but formatting will be lost
- You must #include <iomanip>

SETW()

```
1  // This program displays three rows of numbers.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int num1 = 2897, num2 = 5,    num3 = 837,
8         num4 = 34,   num5 = 7,    num6 = 1623,
9         num7 = 390,  num8 = 3456, num9 = 12;
10
11    // Display the first row of numbers
12    cout << num1 << "  " << num2 << "  " << num3 << endl;
13
14    // Display the second row of numbers
15    cout << num4 << "  " << num5 << "  " << num6 << endl;
16
17    // Display the third row of numbers
18    cout << num7 << "  " << num8 << "  " << num9 << endl;
19    return 0;
20 }
```

- 2897 5 837
- 34 7 1623
- 390 3456 12

SETW()

```
1  // This program displays three rows of numbers.
2  #include <iostream>
3  #include <iomanip>      // Required for setw
4  using namespace std;
5
6  int main()
7  {
8     int num1 = 2897, num2 = 5,    num3 = 837,
9         num4 = 34,   num5 = 7,    num6 = 1623,
10        num7 = 390,  num8 = 3456, num9 = 12;
11
12    // Display the first row of numbers
13    cout << setw(6) << num1 << setw(6)
14         << num2 << setw(6) << num3 << endl;
15
16    // Display the second row of numbers
17    cout << setw(6) << num4 << setw(6)
18         << num5 << setw(6) << num6 << endl;
19
20    // Display the third row of numbers
21    cout << setw(6) << num7 << setw(6)
22         << num8 << setw(6) << num9 << endl;
23    return 0;
24 }
```

2897	5	837
34	7	1623
390	3456	12

SETPRECISION()

- Causes floating-point numbers to be rounded to a certain number of significant digits
- The argument is an integer number representing how many significant digits
- Rounds as you would expect
- Requires #include<iomanip>
- Stays in effect until you call the function again with a different value

SETPRECISION()

```cpp
// This program demonstrates how setprecision rounds a
// floating point value.
#include <iostream>
#include <iomanip>
using namespace std;

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

   quotient = number1 / number2;
   cout << 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;
}
```

SETPRECISION()

- 4.91877
- 4.9188
- 4.919
- 4.92
- 4.9
- 5

- Remember, its significant digits not decimal

Number	Manipulator	Display
28.92786	setprecision(3)	28.9
21	setprecision(5)	21
109.5	setprecision(4)	109.5
34.28596	setprecision(2)	34

SETPRECISION()

```
1  // This program asks for sales figures for 3 days. The total
2  // sales are calculated and displayed in a table.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6
7  int main()
8  {
9     double day1, day2, day3, total;
10
11    // Get the sales for each day.
12    cout << "Enter the sales for day 1: ";
13    cin >> day1;
14    cout << "Enter the sales for day 2: ";
15    cin >> day2;
16    cout << "Enter the sales for day 3: ";
17    cin >> day3;
18
19    // Calculate the total sales.
20    total = day1 + day2 + day3;
```

SETPRECISION()

```
19      // Calculate the total sales.
20      total = day1 + day2 + day3;
21
22      // Display the sales figures.
23      cout << "\nSales Figures\n";
24      cout << "-------------\n";
25      cout << setprecision(5);
26      cout << "Day 1: " << setw(8) << day1 << endl;
27      cout << "Day 2: " << setw(8) << day2 << endl;
28      cout << "Day 3: " << setw(8) << day3 << endl;
29      cout << "Total: " << setw(8) << total << endl;
30      return 0;
31  }
```

- Sales Figures

  Day 1: 321.57
  Day 2: 269.62
  Day 3: 307.77
  Total: 898.96

FIXED

- If setprecision’s argument is too low for a number, it can print it in scientific notation
- It also makes more sense to deal with decimal digits
- Using fixed with setprecision forces it to operate based on decimal digits (fixed point notation)

FIXED

```
1  // This program asks for sales figures for 3 days. The total
2  // sales are calculated and displayed in a table.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6
7  int main()
8  {
9     double day1, day2, day3, total;
10
11    // Get the sales for each day.
12    cout << "Enter the sales for day 1: ";
13    cin >> day1;
14    cout << "Enter the sales for day 2: ";
15    cin >> day2;
16    cout << "Enter the sales for day 3: ";
17    cin >> day3;
18
19    // Calculate the total sales.
20    total = day1 + day2 + day3;
```

FIXED

```
19      // Calculate the total sales.
20      total = day1 + day2 + day3;
21
22      // Display the sales figures.
23      cout << "\nSales Figures\n";
24      cout << "-------------\n";
25      cout << setprecision(2) << fixed;
26      cout << "Day 1: " << setw(8) << day1 << endl;
27      cout << "Day 2: " << setw(8) << day2 << endl;
28      cout << "Day 3: " << setw(8) << day3 << endl;
29      cout << "Total: " << setw(8) << total << endl;
30      return 0;
31  }
```

- Sales Figures

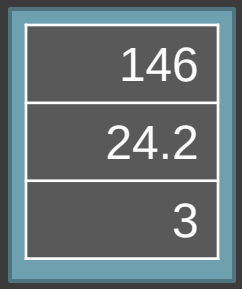
  Day 1: 321.57
  Day 2: 269.62
  Day 3: 307.77
  Total: 898.96

SHOWPOINT

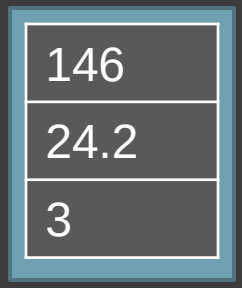
- Even with setprecision(2) and fixed, if a number is an integer then no trailing 0's will be displayed, this is problematic for money
- cout << setprecision(2) << fixed << 23;
  - 23
- cout << setprecision(2) << fixed << showpoint << 23.5;
  - 23.50
- cout << setprecision(6) << showpoint << 23;
  - 23.0000

LEFT AND RIGHT

- By default output is right-justified
  - cout << setw(5) << 146;
    cout << setw(5) << 24.2;
    cout << setw(5) << 3;



- You can use the left manipulator instead
  - cout << left;
    cout << setw(5) << 146;
    cout << setw(5) << 24.2;
    cout << setw(5) << 3;



- This stays in effect until you use the right manipulator

Formatting Input

- Like the cout object, the cin object can also perform formatting
- Recall that character arrays have to be large enough to hold the string + 1 (null terminator)
  - If you store more than it can hold, it will overwrite subsequent data
- You can use setw(), as with cout, to prevent this
  - char word[10];
    cin >> setw(10) >> word;
  - This accounts for the null terminator, allowing only 9 characters to be stored, anything extra is discarded

Reading a Line of Input

- If you've ever tried to cin something with spaces or tabs, you might notice odd results
- If you want to read whitespace data you must use a member function of the cin object
  - cin.getline(variable, size);
  - The first argument is the variable you want to store the string into
  - The second argument is the maximum number of characters you want to read, including the null terminator
  - It will read until size – 1 characters or it encounters the newline character \n (enter)

Reading a Line of Input

```
1  // This program demonstrates cin's getline member function.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     const int SIZE = 81;
8     char sentence[SIZE];
9
10    cout << "Enter a sentence: ";
11    cin.getline(sentence, SIZE);
12    cout << "You entered " << sentence << endl;
13    return 0;
14 }
```

- Enter a sentence: **To be, or not to be, that is the question. [Enter]**
- You entered To be, or not to be, that is the question.

Reading a Character

- Rather than using a character array to store a string, you can store a single character
  - You can use the char type
  - char ch;
    cout << “Type a character and press Enter: ”;
    cin >> ch;
    cout << “You entered “ << ch << endl;
  - This only stores a single character because cin knows that char can only hold one
- This is useful for menu based programs

Reading a Character

- But won't work if you want to implement a "press any key to continue" functionality
  - It requires the user to press enter and you have to type a non-white space character first
- The cin object, by default, ignores whitespace characters, we need something that can accept them
- Just like before, theres a member function
  - cin.get(variable);
  - Reads a single character, including whitespace characters

Reading a Character

```
1  // This program demonstrates cin.get.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     char ch;
8
9     cout << "This program has paused. Press Enter to continue.";
10    cin.get(ch);
11    cout << "Thank you!" << endl;;
12    return 0;
13 }
```

- This program has paused. Press Enter to continue. **[Enter]**
  Thank you!

Mixing cin >> and cin.get()

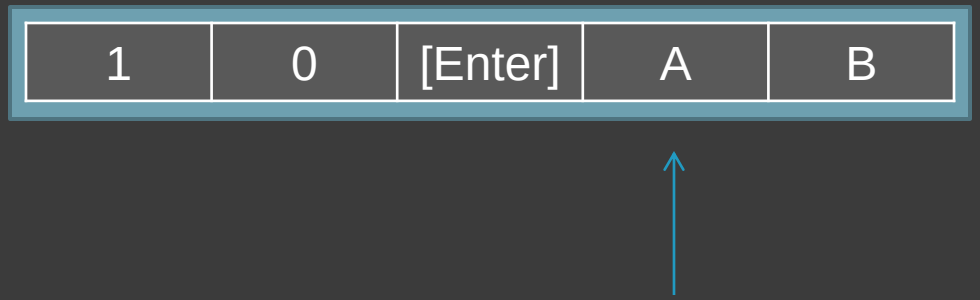
- Be very careful, an unsuspected issue can arise if you are careless
  - char ch;
    int number;
    cout << “Enter a number: “;
    cin >> number;
    cout << “Enter a character: “;
    cin.get(ch);
    cout << “Thank You!”;
    - Enter a number: **100[Enter]**
      Thank You!

Mixing cin >> and cin.get()

- Use a member function to prevent this
  - cin.ignore();
    - char ch;
      int number;
      cout << “Enter a number: “;
      cin >> number;
      cin.ignore();
      cout << “Enter a character: “;
      cin.get(ch);
      cout << “Thank You!”;
      - Enter a number: **100[Enter]**
        Enter a character: **A[Enter]**
        Thank You!

CIN.IGNORE()

- Skips characters in the keyboard buffer
  - cin.ignore();
  - cin.ignore(number);
    - Skips “number” characters
  - cin.ignore(number, character);
    - Skips “number” characters or until the “character” is encountered
    - cin.ignore(20, ‘\n’);



Object-Oriented Programing

- cin is an object
- Objects have their own functions called with the dot (.) operator
  (member functions)
  - cin.getline()
  - cin.get()
  - cin.ignore()
- Member functions allow you to manipulate the data in the object
- The combining of data and functionality within an object is called encapsulation

File Input and Output

- If you want the program to remember or save information, you'll need to use files
- Always three steps
  1. Open the file
     If it does not exist, it will be created
  2. Read / Write data from / to the file
  3. Close the file
     This will cause any changes to be saved

File Input and Output

- #include <fstream>
  - This gives access to three object types
  1. ofstream – output files
  2. ifstream – input files
  3. fstream – files that do both, we won't use this yet
- Create a file object based on one of these types
  - ifstream inputFile;
- Use the member function to open it
  - inputFile.open("customer.dat");

Opening a File

- File location
  - You can specify a file's location manually when you open it
    - inputFile.open("a:\\customer.dat");
      - Note the two backslashes represent one, because of the escape characters
    - Alternatively, you can open the file as on the previous page, as long as the file is stored within the project's self-named folder (for visual studio)
      - VisualStudio\Projects\TestProj\TestProj\customer.dat
      - inputFile.open("customer.dat");

Opening a File

- You can also allow the user to specify the filename as well
  - ifstream inputFile;
    char fileName[20];
    cout << “Enter the name of the file: “;
    cin >> fileName;
    inputFile.open(fileName);
- You can also combine the object creation and open member function into one statement
  - ifstream inputFile(“customer.dat”);

Closing a File

- Changes are not saved until the file is closed
- Operating systems limit the number of files that can be opened
- The software can slow down if too many files are open as well
- Simply use the member function to close it
  - outputFile.close();
  - No more data can be writen to outputFile until it is reopened

Writing Data to a File

- As simple as using cout, but you use the output file object's name instead
  - outputFile << "Some text";
  - outputFile << "Price: " << price;
  - outputFile << "Line 1" << endl;
    outputFile << "Line 2\n";
    outputFile << "Line 3";
  - outputFile << 25;
    outputFile << 26 << 27 << 28;

Writing Data to a File

```
1  // This program writes data to a file.
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5
6  int main()
7  {
8     ofstream outputFile;
9     outputFile.open("demofile.txt");
10
11    cout << "Now writing information to the file.\n";
12
13    // Write 4 great names to the file
14    outputFile << "Bach\n";
15    outputFile << "Beethoven\n";
16    outputFile << "Mozart\n";
17    outputFile << "Schubert\n";
18
19    // Close the file
20    outputFile.close();
21    cout << "Done.\n";
22    return 0;
23 }
```

Reading Data from a File

- As before, it is like using cin but with the file object's name

```
1  // This program reads information from a file.
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5
6  int main()
7  {
8     ifstream inFile;
9     const int SIZE = 81;
10    char name[SIZE];
11
12    inFile.open("demofile.txt");
```

Reading Data from a File

```
12      inFile.open("demofile.txt");
13      cout << "Reading information from the file.\n\n";
14
15      inFile >> name;           // Read name 1 from the file
16      cout << name << endl;     // Display name 1
17
18      inFile >> name;           // Read name 2 from the file
19      cout << name << endl;     // Display name 2
20
21      inFile >> name;           // Read name 3 from the file
22      cout << name << endl;     // Display name 3
23
24      inFile >> name;           // Read name 4 from the file
25      cout << name << endl;     // Display name 4
26
27      inFile.close();           // Close the file
28      cout << "\nDone.\n";
29      return 0;
30  }
```

Reading and Writing Notes

- By default, data is read from an input file sequentially beginning at the top
- If the file does not exist, an error is thrown

- By default, when you open a output file for writing, its previous data is erased
- If the file does not exist, it will be created

- In later chapters you will learn how to override these default behaviors

Reading and Writing Notes

- Be careful when reading or writing data
  - The default behaviors of the cin and cout objects apply, specifically whitespace
  - istream inputFile("numbers.txt");
    int num1, num2, num3;
    cout << num1 << endl;
    cout << num2 << endl;
    cout << num3 << endl;
    - 123
      blank
      blank