**THE CIN OBJECT**

- Standard input object
- Like `cout`, requires `iostream` file
- Used to read input from keyboard
- Information retrieved from `cin` with `>>`
- Input is stored in one or more variables

- `cin` converts data to the type that matches the variable:

```cpp
int height;
cout << "How tall is the room? ";
cin >> height;
```
THE CIN OBJECT

IN PROGRAM

Program 3-1

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

Program Output with Example Input Shown in Bold

This program calculates the area of a rectangle.
What is the length of the rectangle? 10 [Enter]
What is the width of the rectangle? 20 [Enter]
The area of the rectangle is 200.

THE CIN OBJECT

DISPLAYING A PROMPT

• A prompt is a message that instructs the user to enter data.
• You should always use cout to display a prompt before each cin statement.
  
  cout << "How tall is the room? ";
  cin >> height;

• Can be used to input more than one value:
  
  cin >> height >> width;

• Multiple values from keyboard must be separated by spaces
• Order is important: first value entered goes to first variable, etc.
THE CIN OBJECT
GATHER MULTIPLE VALUES

Program 3.2

```cpp
// This program asks the user to enter the length and width of a rectangle. It calculates the rectangle's area and displays // the value on the screen.
#include <iostream>
using namespace std;

int main()
{
    int length, width, area;
    cout << "This program calculates the area of a rectangle. Enter the length and width of the rectangle: ";
    cin >> length >> width;
    area = length * width;
    cout << "The area of the rectangle is: " << area; // << endl;
    return 0;
}
```

Program Output with Example Input Shown in Bold

This program calculates the area of a rectangle. Enter the length and width of the rectangle separated by a space. **10 20** (Enter) The area of the rectangle is 200

THE CIN OBJECT
READS DIFFERENT DATA TYPES

Program 3.3

```cpp
// This program demonstrates how cin can read multiple values of different data types.
#include <iostream>
using namespace std;

int main()
{
    int whole;
    double fractional;
    char letter;
    cout << "Enter an integer, a double, and a character: ";
    cin >> whole >> fractional >> letter;
    cout << "Whole: " << whole; // << endl;
    cout << "Fractional: " << fractional; // << endl;
    cout << "Letter: " << letter; // << endl;
    return 0;
}
```

Program Output with Example Input Shown in Bold

Enter an integer, a double, and a character: **4 5.7 b** (Enter) Whole: 4 Fractional: 5.7 Letter: b
MATHEMATICAL EXPRESSIONS

• Can create complex expressions using multiple mathematical operators
• An expression can be a literal, a variable, or a mathematical combination of constants and variables
• Can be used in assignment, cout, other statements:

  area = 2 * PI * radius;
  cout << "border is: " << 2*(l+w);

ORDER OF OPERATIONS

• In an expression with more than one operator, evaluate in this order:
  - (unary negation), in order, left to right
  * / %, in order, left to right
  + -, in order, left to right

• In the expression \(2 + 2 * 2 - 2\)
  - Evaluate second
  - Evaluate first
  - Evaluate third

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 + 2 * 4</td>
<td>13</td>
</tr>
<tr>
<td>10 / 2 - 3</td>
<td>2</td>
</tr>
<tr>
<td>8 + 12 * 2 - 4</td>
<td>28</td>
</tr>
<tr>
<td>4 + 17 % 2 - 1</td>
<td>4</td>
</tr>
<tr>
<td>6 - 3 * 2 + 7 - 1</td>
<td>6</td>
</tr>
</tbody>
</table>
MATHEMATICAL EXPRESSIONS

ASSOCIATIVITY OF OPERATORS

- (unary negation) associates right to left
- *, /, %, +, - associate right to left
- parentheses ( ) can be used to override the order of operations:

\[
\begin{align*}
2 + 2 & \times 2 - 2 = 4 \\
(2 + 2) & \times 2 - 2 = 6 \\
2 + 2 & \times (2 - 2) = 2 \\
(2 + 2) & \times (2 - 2) = 0
\end{align*}
\]

Table 3-4  More Simple Expressions and Their Values

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5 + 2) * 4</td>
<td>28</td>
</tr>
<tr>
<td>10 / (5 - 3)</td>
<td>5</td>
</tr>
<tr>
<td>8 + 12 * (6 - 2)</td>
<td>56</td>
</tr>
<tr>
<td>(4 + 17) % 2 - 1</td>
<td>0</td>
</tr>
<tr>
<td>(6 - 3) * (2 + 7) / 3</td>
<td>9</td>
</tr>
</tbody>
</table>
MATHEMATICAL EXPRESSIONS

ALGEBRAIC EXPRESSIONS

• Multiplication requires an operator:
  \[ \text{Area}=lw \] is written as \[ \text{Area} = l \times w; \]

• There is no exponentiation operator:
  \[ \text{Area}=s^2 \] is written as \[ \text{Area} = \text{pow}(s, 2); \]

• Parentheses may be needed to maintain order of operations:
  \[ m = \frac{y_2 - y_1}{x_2 - x_1} \]
  is written as \[ m = (y_2-y_1) / (x_2-x_1); \]

<table>
<thead>
<tr>
<th>Table 3-5 Algebraic and C++ Multiplication Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebraic Expression</td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>(6B)</td>
</tr>
<tr>
<td>((3)(12))</td>
</tr>
<tr>
<td>(4xy)</td>
</tr>
</tbody>
</table>

TYPE CONVERSION

WHEN YOU MIX APPLES WITH ORANGES:

• Operations are performed between operands of the same type.
• If not of the same type, C++ will convert one to be the type of the other
• This can impact the results of calculations.
TYPE CONVERSION

HIERARCHY OF TYPES

• Highest:
  long double
double
float
unsigned long
long
unsigned int
int

• Lowest:
• Ranked by largest number they can hold

TYPE CONVERSION

TYPE COERCION

• **Type Coercion**: automatic conversion of an operand to another data type
• **Promotion**: convert to a higher type
• **Demotion**: convert to a lower type

• **Coercion Rules**

  1. char, short, unsigned short automatically promoted to int

  2. When operating on values of different data types, the lower one is promoted to the type of the higher one.

  3. When using the – operator, the type of expression on right will be converted to type of variable on left
OVERFLOW AND UNDERFLOW

• Occurs when assigning a value that is too large (overflow) or too small (underflow) to be held in a variable

• Variable contains value that is ‘wrapped around’ set of possible values

• Different systems may display a warning/error message, stop the program, or continue execution using the incorrect value

TYPE CASTING

• Used for manual data type conversion

• Useful for floating point division using ints:

```cpp
double m;

m = static_cast<double>(y2-y1)/(x2-x1);
```

• Useful to see int value of a char variable:

```cpp
char ch = 'C';
cout << ch << " is "
    << static_cast<int>(ch);
```
TYPE CASTING

IN PROGRAM

Program 3-9

```cpp
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     cout << "How many books do you plan to read? ";
11     cin >> books;
12     cout << "How many months will it take you to read them? ";
13     cin >> months;
14     perMonth = static_cast<double>(books) / months;
15     cout << "That is " << perMonth << " books per month.\n\n";
16     return 0;
17  }
```

Program Output with Example Input Shown in Bold

How many books do you plan to read? 30 [Enter]
How many months will it take you to read them? 7 [Enter]
That is 4.28571 books per month.

TYPE CASTING

C-STYLE AND PRESTANDARD TYPE CAST EXPRESSIONS

• C-Style cast: data type name in ()
  `cout << ch << " is " << (int)ch;`

• Prestandard C++ cast: value in ()
  `cout << ch << " is " << int(ch);`

• Both are still supported in C++, although `static_cast` is preferred
MULTIPLE ASSIGNMENT AND COMBINED ASSIGNMENT

- The `=` can be used to assign a value to multiple variables:
  \[ x = y = z = 5; \]

- Value of `=` is the value that is assigned

- Associates right to left:
  \[ x = (y = (z = 5)); \]

<table>
<thead>
<tr>
<th>Statement</th>
<th>What It Does</th>
<th>Value of x After the Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = x + 4; )</td>
<td>Adds 4 to x</td>
<td>10</td>
</tr>
<tr>
<td>( x = x - 3; )</td>
<td>Subtracts 3 from x</td>
<td>3</td>
</tr>
<tr>
<td>( x = x \times 10; )</td>
<td>Multiplies x by 10</td>
<td>60</td>
</tr>
<tr>
<td>( x = x / 2; )</td>
<td>Divides x by 2</td>
<td>3</td>
</tr>
<tr>
<td>( x = x % 4 )</td>
<td>Makes x the remainder of x / 4</td>
<td>2</td>
</tr>
</tbody>
</table>

COMBINED ASSIGNMENT

- Look at the following statement:
  \[ \text{sum} = \text{sum} + 1; \]

- This adds 1 to the variable sum.
MULTIPLE ASSIGNMENT AND COMBINED ASSIGNMENT

COMBINED ASSIGNMENT

- The combined assignment operators provide a shorthand for these types of statements.
- The statement
  \[
  \text{sum} = \text{sum} + 1;
  \]
  - is equivalent to
  \[
  \text{sum} += 1;
  \]

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example Usage</th>
<th>Equivalent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>x += 5;</td>
<td>x = x + 5;</td>
</tr>
<tr>
<td>-=</td>
<td>y -= 2;</td>
<td>y = y - 2;</td>
</tr>
<tr>
<td>*=</td>
<td>z *= 10;</td>
<td>z = z * 10;</td>
</tr>
<tr>
<td>/=</td>
<td>a /= b;</td>
<td>a = a / b;</td>
</tr>
<tr>
<td>%=</td>
<td>c %= 3;</td>
<td>c = c % 3;</td>
</tr>
</tbody>
</table>

FORMATTING OUTPUT

- Can control how output displays for numeric, string data:
  - size
  - position
  - number of digits
- Requires `iomanip` header file
FORMATTING OUTPUT
STREAM MANIPULATORS

• Used to control how an output field is displayed

• Some affect just the next value displayed:
  • `setw(x)`: print in a field at least x spaces wide.
  • Use more spaces if field is not wide enough

### Program 3-13
```cpp
#include <iostream>
// Required for setw
using namespace std;

int main()
{
    int num1 = 2897, num2 = 5, num3 = 837,
    num4 = 34, num5 = 7, num6 = 1623,
    num7 = 390, num8 = 3456, num9 = 12;

    // Display the first row of numbers
    cout << setw(6) << num1 << setw(6)
         << num2 << setw(6) << num3 << endl;

    // Display the second row of numbers
    cout << setw(6) << num4 << setw(6)
         << num5 << setw(6) << num6 << endl;

    // Display the third row of numbers
    cout << setw(6) << num7 << setw(6)
         << num8 << setw(6) << num9 << endl;
    return 0;
}
```

**Program Output**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2897</td>
<td>5</td>
<td>837</td>
</tr>
<tr>
<td>34</td>
<td>7</td>
<td>1623</td>
</tr>
<tr>
<td>390</td>
<td>3456</td>
<td>12</td>
</tr>
</tbody>
</table>

FORMATTING OUTPUT
STREAM MANIPULATORS

• Some affect values until changed again:
  • **fixed**: use decimal notation for floating-point values
  • **setprecision(x)**: when used with fixed, print floating-point value using x digits after the decimal. Without fixed, print floating-point value using x significant digits
  • **showpoint**: always print decimal for floating-point values
FORMATTING OUTPUT

IN PROGRAM

Program 3-17

```cpp
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;
21     cout << "\nSales Figures\n-------------------\nDay 1: \1321.87 [Enter]
Day 2: \1869.26 [Enter]
Day 3: \1403.77 [Enter]
Total: 4594.90```
FORMATTING OUTPUT
STREAM MANIPULATORS

Table 3-12

<table>
<thead>
<tr>
<th>Stream Manipulator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setw(n)</td>
<td>Establishes a print field of n spaces.</td>
</tr>
<tr>
<td>fixed</td>
<td>Displays floating-point numbers in fixed point notation.</td>
</tr>
<tr>
<td>showpoint</td>
<td>Causes a decimal point and trailing zeroes to be displayed, even if there is no fractional part.</td>
</tr>
<tr>
<td>setprecision(n)</td>
<td>Sets the precision of floating-point numbers.</td>
</tr>
<tr>
<td>left</td>
<td>Causes subsequent output to be left justified.</td>
</tr>
<tr>
<td>right</td>
<td>Causes subsequent output to be right justified.</td>
</tr>
</tbody>
</table>

WORKING WITH CHARACTERS AND STRING OBJECTS
GETLINE

- Using **cin** with the `>>` operator to input strings can cause problems:

- It passes over and ignores any leading **whitespace characters** (spaces, tabs, or line breaks)

- To work around this problem, you can use a C++ function named **getline**.
WORKING WITH CHARACTERS AND STRING OBJECTS

USING GETLINE IN PROGRAM

Program 3-19

```cpp
1 // This program demonstrates using the getline function
2 // to read character data into a string object.
3 #include <iostream>
4 #include <string>
5 using namespace std;
6
7 int main()
8 {
9     string name;
10    string city;
11    cout << "Please enter your name: ";
12    getline(cin, name);
13    cout << "Enter the city you live in: ";
14    getline(cin, city);
15    cout << "Hello, " << name << endl;
16    cout << "You live in " << city << endl;
17    return 0;
18 }
```

Program Output with Example Input Shown in Bold

Please enter your name: Kate Smith [Enter]
Enter the city you live in: Raleigh [Enter]
Hello, Kate Smith
You live in Raleigh

WORKING WITH CHARACTERS AND STRING OBJECTS

• To read a single character:
  • Use `cin`:

```cpp
    char ch;
    cout << "Strike any key to continue";
    cin >> ch;
```

Problem: will skip over blanks, tabs, <CR>

• Use `cin.get()`:

```cpp
    cin.get(ch);
```

Will read the next character entered, even whitespace
WORKING WITH CHARACTERS AND STRING OBJECTS

USING **CIN.GET()** IN PROGRAM

**Program 3-21**

```cpp
1 // This program demonstrates three ways
2 // to use cin.get() to pause a program.
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8     char ch;
9     cout << "This program has paused. Press Enter to continue.\n";
10    cin.get(ch);
11    cout << "It has paused a second time. Please press Enter again.\n";
12    ch = cin.get();
13    cout << "It has paused a third time. Please press Enter again.\n";
14    cin.get();
15    cout << "Thank you!\n";
16    return 0;
17 }
```

**Program Output with Example Input Shown in Bold**

This program has paused. Press Enter to continue. [Enter]
It has paused a second time. Please press Enter again. [Enter]
It has paused a third time. Please press Enter again. [Enter]
Thank you!

- Mixing **cin >>** and **cin.get()** in the same program can cause input errors that are hard to detect.

- To skip over unneeded characters that are still in the keyboard buffer, use **cin.ignore()**:

  ```cpp
cin.ignore(); // skip next char
  cin.ignore(10, '\n'); // skip the next
           // 10 char. or until a '\n'
  ```
WORKING WITH CHARACTERS AND STRING OBJECTS

STRING MEMBER FUNCTIONS AND OPERATORS

- To find the length of a string:
  ```
  string state = "Texas";
  int size = state.length();
  ```

- To concatenate (join) multiple strings:
  ```
  greeting2 = greeting1 + name1;
  greeting1 = greeting1 + name2;
  ```
  Or using the `+=` combined assignment operator:
  ```
  greeting1 += name2;
  ```

MORE MATHEMATICAL LIBRARY FUNCTIONS

- Require `cmath` header file
- Take `double` as input, return a `double`
- Commonly used functions:
  ```
  sin  Sine
  cos  Cosine
  tan  Tangent
  sqrt Square root
  log  Natural (e) log
  abs  Absolute value (takes and returns an int)
  ```

- These require `cstdlib` header file
- `rand()`: returns a random number (int) between 0 and the largest `int` the compute holds. Yields same sequence of numbers each time program is run.
- `srand(x)`: initializes random number generator with unsigned `int x`
HAND TRACING A PROGRAM

- Hand trace a program: act as if you are the computer, executing a program:
  - step through and 'execute' each statement, one-by-one
  - record the contents of variables after statement execution, using a hand trace chart (table)

- Useful to locate logic or mathematical errors

---

HAND TRACING A PROGRAM

IN PROGRAM

```cpp
Program 3-26 (with hand trace chart filled)

1 // This program asks for three numbers, then
2 // displays the average of the numbers.
3 #include <iostream>
4 using namespace std;
5 int main()
6 {
7   double num1, num2, num3, avg;
8   cout << "Enter the first number: ";
9   cin >> num1;
10  cout << "Enter the second number: ";
11  cin >> num2;
12  cout << "Enter the third number: ";
13  cin >> num3;
14  avg = num1 + num2 + num3 / 3;
15  cout << "The average is " << avg << endl;
16  return 0;
```

---

HAND TRACING A PROGRAM

IN PROGRAM

<table>
<thead>
<tr>
<th>num1</th>
<th>num2</th>
<th>num3</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
</tr>
</tbody>
</table>
A CASE STUDY

• General Crates, Inc. builds custom-designed wooden crates.
• You have been asked to write a program that calculates the:
  • Volume (in cubic feet)
  • Cost
  • Customer price
  • Profit of any crate GCI builds

VARIABLES

Table 3-14

<table>
<thead>
<tr>
<th>Constant or Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COST_PER_CUBIC FOOT</td>
<td>A named constant, declared as a double and initialized with the value 0.25. This represents the cost to build a crate, per cubic foot.</td>
</tr>
<tr>
<td>CHARGE_PER_CUBIC FOOT</td>
<td>A named constant, declared as a double and initialized with the value 0.5. This represents the amount charged for a crate, per cubic foot.</td>
</tr>
<tr>
<td>length</td>
<td>A double variable to hold the length of the crate, which is input by the user.</td>
</tr>
<tr>
<td>width</td>
<td>A double variable to hold the width of the crate, which is input by the user.</td>
</tr>
<tr>
<td>height</td>
<td>A double variable to hold the height of the crate, which is input by the user.</td>
</tr>
<tr>
<td>volume</td>
<td>A double variable to hold the volume of the crate. The value stored in this variable is calculated.</td>
</tr>
<tr>
<td>cost</td>
<td>A double variable to hold the cost of building the crate. The value stored in this variable is calculated.</td>
</tr>
<tr>
<td>charge</td>
<td>A double variable to hold the amount charged to the customer for the crate. The value stored in this variable is calculated.</td>
</tr>
<tr>
<td>profit</td>
<td>A double variable to hold the profit GCI makes from the crate. The value stored in this variable is calculated.</td>
</tr>
</tbody>
</table>
A Case Study

Program Design

- The program must perform the following general steps:

  - Step 1:
    - Ask the user to enter the dimensions of the crate

  - Step 2:
    - Calculate:
      - the crate’s volume
      - the cost of building the crate
      - the customer’s charge
      - the profit made

  - Step 3:
    - Display the data calculated in Step 2.

---

Figure 3.7

```
<table>
<thead>
<tr>
<th>Calculate Crate Volume, Cost, Price, and Profit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Crate Dimensions.</td>
</tr>
<tr>
<td>Calculate Volume, Cost, Customer Charge, and Profit.</td>
</tr>
<tr>
<td>Display Calculated Data.</td>
</tr>
</tbody>
</table>
```
A CASE STUDY
GET CRATE DIMENSIONS

Figure 3-8

Get Crate Dimensions.

- Get Length.
- Get Width.
- Get Height.

A CASE STUDY
CALCULATE VOLUME, COST, CUSTOMER CHARGE, AND PROFIT

Figure 3-9

Calculate Volume, Cost, Customer Charge, and Profit.

- Calculate the Crate's Volume.
- Calculate the Crate's Cost.
- Calculate the Customer Charge.
- Calculate the Profit Made.
A CASE STUDY
DISPLAY CALCULATED DATA

Figure 3.10

- Display Calculated Data.
  - Display the Crate's Volume.
  - Display the Crate's Cost.
  - Display the Customer Charge.
  - Display the Profit Made.

A CASE STUDY
PSEUDOCODE

• Ask the user to input the crate's length.
• Ask the user to input the crate's width.
• Ask the user to input the crate's height.
• Calculate the crate's volume.
• Calculate the cost of building the crate.
• Calculate the customer's charge for the crate.
• Calculate the profit made from the crate.
• Display the crate's volume.
• Display the cost of building the crate.
• Display the customer's charge for the crate.
• Display the profit made from the crate.
A CASE STUDY

CALCULATIONS

- The following formulas will be used to calculate the crate's volume, cost, charge, and profit:
  
  - volume = length × width × height
  - cost = volume × 0.23
  - charge = volume × 0.5
  - profit = charge − cost

---

THE PROGRAM

Program 3-27

```cpp
// This program is used by General Crates, Inc. to calculate
// the volume, cost, customer charge, and profit of a crate
// of any size. It calculates this data from user input, which
// consists of the dimensions of the crate.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    // Constants for cost and amount charged
    const double COST_PER_CUBIC_FOOT = 0.23;
    const double CHARGE_PER_CUBIC_FOOT = 0.5;

    // Variables
    double length, // The crate's length
           width, // The crate's width
           height, // The crate's height
           volume, // The volume of the crate
           cost, // The cost to build the crate
           charge, // The customer charge for the crate
           profit; // The profit made on the crate

    // Set the desired output formatting for numbers.
    cout << setprecision(2) << fixed << showpoint;

    // The rest of the program...
```
// Prompt the user for the crate's length, width, and height
    cout << "Enter the dimensions of the crate (in feet):\n";
    cout << "Length: ";
    cin >> length;
    cout << "Width: ";
    cin >> width;
    cout << "Height: ";
    cin >> height;

    // Calculate the crate's volume, the cost to produce it,
    // the charge to the customer, and the profit.
    volume = length * width * height;
    cost = volume * COST_PER_CUBIC_FOOT;
    charge = volume * CHARGE_PER_CUBIC_FOOT;
    profit = charge - cost;

    // Display the calculated data.
    cout << "The volume of the crate is ";
    cout << volume << " cubic feet.\n";
    cout << "Cost to build: $" << cost << endl;
    cout << "Charge to customer: $" << charge << endl;
    cout << "Profit: $" << profit << endl;
    return 0;
}