# 1.3 Case Study: Revenue Calculation

## PROBLEM

Sam Splicer installs coaxial cable for the Metro Cable Company. For each installation, there is a basic service charge of $25.00 and an additional charge of $2.00 for each foot of cable. The president of the cable company would like a program to compute the revenue generated by Sam in any given month. For example, if during the month of January, Sam installs a total of 263 yards of cable at 27 different locations, he generates $2253.00 in revenue.

## OBJECT-CENTERED DESIGN

Following the principles of object-centered design, we describe the behavior of the program, identify the objects and operations, and then write an algorithm.

**BEHAVIOR.** The program should display on the screen a prompt for the number of installations performed and the total number of yards of cable installed. The user should enter those values from the keyboard. The program should compute and display on the screen the total amount of revenue resulting from these installations.

**OBJECTS.** From our behavioral description, we can identify the following objects in this problem:

| | Software Objects | | |
|---|---|---|---|
| **Problem Objects** | **Type** | **Kind** | **Name** |
| screen | `ostream` | variable | `cout` |
| prompt | `string` | constant | none |
| number of installations | `int` | variable | *installations* |
| total number of yards of cable | `double` | variable | *yardsOfCable* |
| keyboard | `istream` | variable | `cin` |
| total amount of revenue | `double` | variable | *totalRevenue* |

**OPERATIONS.** From our behavioral description, we can identify the following operations:

i. Display a prompt for installations and yards of cable to `cout`
ii. Read an integer from `cin` into *installations*
iii. Read a real from `cin` into *yardsOfCable*
iv. Compute *totalRevenue* from *installations* and *yardsOfCable*
v. Display *totalRevenue* to `cout` (with some descriptive text)

Each of these operations is predefined in C++, except for computing *totalRevenue* from *installations* and *yardsOfCable.* Using information from the original problem description, we can break this step down into simpler operations:

Compute *totalRevenue* from *installations* and *yardsOfCable*
Compute *installRevenue* = 25.00 * *installations*
Compute *feetOfCable* = 3 * *yardsOfCable*
Compute *cableRevenue* = 2.00 * *feetOfCable*
Compute *totalRevenue* = *installRevenue* + *cableRevenue*

This refinement of the computation adds six new objects to our list:

| | Software Objects | | |
|---|---|---|---|
| **Probem Objects** | **Type** | **Kind** | **Name** |
| screen | `ostream` | varying | `cout` |
| prompt | `string` | constant | none |
| number of installations | `int` | varying | *installations* |
| total number of yards of cable | `double` | varying | *yardsOfCable* |
| keyboard | `istream` | varying | `cin` |
| total amount of revenue | `double` | varying | *totalRevenue* |
| installation revenue | `double` | varying | *installRevenue* |
| installation charge (25.00) | `double` | constant | *INSTALL_CHARGE* |
| feet of cable | `double` | varying | *feetOfCable* |
| number of feet in a yard (3) | `int` | constant | none |
| cable revenue | `double` | varying | *cableRevenue* |
| cable cost per foot (2.00) | `double` | constant | *CABLE_COST_PER_FOOT* |

We give each of these new objects a name except for the number of feet in a yard, because it is a fixed quantity that will not change in the future, unlike the installation charge and the cable cost per foot, which may change.

**ALGORITHM.** Given our objects and operations, we can order them into the following algorithm:

### Algorithm:

1. Display a prompt for the number of installations and the total number of yards of cable to `cout`.
2. Read *installations* and *yardsOfCable* from `cin`.
3. Compute *installRevenue* = *INSTALL_CHARGE* * *installations*.
4. Compute *feetOfCable* = 3 * *yardsOfCable.*

5. Compute *cableRevenue* = *CABLE_COST_PER_FOOT* * *feetOfCable*.
6. Compute *totalRevenue* = *installRevenue* + *cableRevenue.*
7. Display *totalRevenue* to `cout`.

## CODING IN C++

Once we have an algorithm to serve as our blueprint, we can encode it in C++ as shown below. Constant objects like `INSTALL_CHARGE` and `CABLE_COST_PER_FOOT`, whose values are likely to change in the future, are defined at the beginning of the program, because this makes them easy to find when an update is necessary.

**Case Study 1.3-1** Revenue Computation.

```
/* revenue.cpp computes revenues for a cable installer.
 *
 * Input:  Number of installations, yards of cable installed
 * Output: The total revenue resulting from these installations
 ************************************************************/

#include <iostream>                    // cin, cout, <<, >>
using namespace std;

int main()
{
   const double INSTALL_CHARGE = 25.00;
   const double CABLE_COST_PER_FOOT = 2.00;

   cout << "Enter the number of installations\n"
        << "followed by the total yards of cable installed: ";
   int installations;
   double yardsOfCable;
   cin >> installations >> yardsOfCable;

   double installRevenue = INSTALL_CHARGE * installations;
   double feetOfCable = 3.0 * yardsOfCable;
   double cableRevenue = CABLE_COST_PER_FOOT * feetOfCable;
   double totalRevenue = installRevenue + cableRevenue;

   cout << "\nThe total revenue for these installations is $"
        << totalRevenue << endl;
}
```

## TESTING, EXECUTION, AND DEBUGGING

Once our program compiles without errors, we check that it is performing correctly by executing it using sample data:

```
Enter the number of installations,
followed by the total yards of cable installed: 1 1.0

The total revenue for these installations is $31
```

This data has been chosen because it is easy to verify the result. One installation results in an installation revenue of $25.00, and 1.0 yard of cable is 3.0 feet of cable, which at 2.00 per foot results in a cable revenue of $6.00. Because $25.00 + $6.00 = $31.00, our program performed correctly in this case.

We try another set of values for which it is easy to verify the result:

```
Enter the number of installations,
followed by the total yards of cable installed: 2 2.25

The total revenue for these installations is $63.5
```

The installation revenue is $25.00 * 2 = $50.00; 2.25 yards of cable is 6.75 feet of cable, which at 2.00 per foot gives a cable revenue of $13.50; and $50.00 + $13.50 = $63.50, which agrees with the output produced by our program. Testing with data sets like these increases our confidence in the correctness of our program, and we can run it with the input data given in the statement of the problem:

```
Enter the number of installations,
followed by the total yards of cable installed: 27 263.5

The total revenue for these installations is $2256
```

## PROGRAM MAINTENANCE

Suppose that sometime after the program in Figure Case Study was written, the Metro Cable Company raised the basic service charge for an installation from $25.00 to $30.00. Obviously, the program will no longer compute the correct revenue, so it must be modified to reflect the change. In the program in Case Study 1.3-1, this is easily done by changing the line

```
const double INSTALL_CHARGE = 25.00;
```

to

```
const double INSTALL_CHARGE = 30.00;
```

and then recompiling the program. The program will then calculate `installRevenue` using the new value of `INSTALL_CHARGE`, so that its output will reflect the new service charge.

As a second example of program maintenance, suppose that the president of the Metro Cable Company requests that exactly two decimal digits be displayed for revenue, since this is customary for monetary values. Performing this upgrade involves formatting the value of `totalRevenue` before we display it. This in turn requires a better understanding of the C++ input/output system, which is discussed in Chapters 5 and 9. Until then, the following program shows how it can be done:[1]

**Case Study 1.3-2** Revenue Computation—Revised.

```
/* revenue.cpp computes revenues for a cable installer.
 *
 * Input:  Number of installations, yards of cable installed
 * Output: The total revenue resulting from these installations
 ************************************************************/
#include <iostream>      // cin, cout, <<, >>, fixed, showpoint
#include <iomanip>       // setprecision()
using namespace std;

int main()
{
   const double INSTALL_CHARGE = 30.00;
   const double CABLE_COST_PER_FOOT = 2.00;

   cout << "Enter the number of installations\n"
        << "followed by the total yards of cable installed: ";
   int installations;
   double yardsOfCable;
   cin >> installations >> yardsOfCable;

   double installRevenue = INSTALL_CHARGE * installations;
   double feetOfCable = 3.0 * yardsOfCable;
   double cableRevenue = CABLE_COST_PER_FOOT * feetOfCable;
   double totalRevenue = installRevenue + cableRevenue;
```

1. If your compiler is not fully ANSI-C++ compliant, you may have to replace the line

   ```
   << fixed << showpoint
   ```

   by

   ```
   << setiosflags (ios::fixed | ios::showpoint)
   ```

   Also, `<iomanip.h>` may be needed instead of `<iomanip>` — see Footnote 1 of Chap. 1 in the text.

```
   cout << "\nThe total revenue for these installations is $"
        << fixed                        // use fixed-point form,
        << showpoint                    // show decimal point,
        << setprecision(2)              // and 2 decimal places
        << totalRevenue << endl;
}
```

**Sample run:**

```
Enter the number of installations,
followed by the total yards of cable installed: 1 1.0

The total revenue for these installations is $36.00
...
Please enter the number of installations,
followed by the total yards of cable installed: 2 2.25

The total revenue for these installations is $73.50
```