Programs and Programming Languages

AAA

Monday 1/11/2016

CONCEPT: A program is a set of instructions a computer follows in order to perform a task. A programming language is a special language used to write computer programs.

What Is a Program?

p8

Computers are designed to follow instructions. A computer program is a set of instructions that tells the computer how to solve a problem or perform a task. For example, suppose we want the computer to calculate someone's gross pay. Here is a list of things the computer should do:

1. Display a message on the screen asking "How many hours did you work?"
2. Wait for the user to enter the number of hours worked. Once the user enters a number, store it in memory.
3. Display a message on the screen asking "How much do you get paid per hour?"
4. Wait for the user to enter an hourly pay rate. Once the user enters a number, store it in memory.
5. Multiply the number of hours by the amount paid per hour, and store the result in memory.
6. Display a message on the screen that tells the amount of money earned. The message must include the result of the calculation performed in Step 5.

Collectively, these instructions are called an *algorithm*. An algorithm is a set of well-defined steps for performing a task or solving a problem. Notice these steps are sequentially ordered. Step 1 should be performed before Step 2, and so forth. It is important that these instructions be performed in their proper sequence.

Figure 1-9

Program 1-1

p9

```
// This program calculates the user's pay.
#include <iostream>
using namespace std;

int main()
{
   double hours, rate, pay;

   // Get the number of hours worked.
   cout << "How many hours did you work? ";
   cin >> hours;

   // Get the hourly pay rate.
   cout << "How much do you get paid per hour? ";
   cin >> rate;

   // Calculate the pay.
   pay = hours * rate;

   // Display the pay.
   cout << "You have earned $" << pay << endl;
   return 0;
}
```

Program Output with Example Input Shown in Bold

```
How many hours did you work? 10 [Enter]
How much do you get paid per hour? 15 [Enter]
You have earned $150
```

Figure 1-9

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools such as a hierarchy chart, flowcharts, or pseudocode to create a model of the program.
4. Check the model for logical errors.
5. Type the code, save it, and compile it.
6. Correct any errors found during compilation. Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any errors found while running the program. Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

p19

The steps listed in Figure 1-9 emphasize the importance of planning. Just as there are good ways and bad ways to paint a house, there are good ways and bad ways to create a program. A good program always begins with planning.

BBB

Monday 1/11/2016

With the pay-calculating program as our example, let's look at each of the steps in more detail.

p19

1. Clearly define what the program is to do.

This step requires that you identify the purpose of the program, the information that is to be input, the processing that is to take place, and the desired output. Let's examine each of these requirements for the example program:

Purpose	To calculate the user's gross pay.
Input	Number of hours worked, hourly pay rate.
Process	Multiply number of hours worked by hourly pay rate. The result is the user's gross pay.
Output	Display a message indicating the user's gross pay.

2. Visualize the program running on the computer.

Before you create a program on the computer, you should first create it in your mind. Step 2 is the visualization of the program. Try to imagine what the computer screen looks like while the program is running. If it helps, draw pictures of the screen, with sample input and output, at various points in the program. For instance, here is the screen produced by the pay-calculating program:

```
How many hours did you work? 10
How much do you get paid per hour? 15
You have earned $150
```

In this step, you must put yourself in the shoes of the user. What messages should the program display? What questions should it ask? By addressing these concerns, you will have already determined most of the program's output.

Predict the Result

Questions 33–35 are programs expressed as English statements. What would each display on the screen if they were actual programs?

33. The variable x starts with the value 0.
    The variable y starts with the value 5.
    Add 1 to x.
    Add 1 to y.
    Add x and y, and store the result in y.
    Display the value in y on the screen.

p26

34. The variable j starts with the value 10.
    The variable k starts with the value 2.
    The variable l starts with the value 4.
    Store the value of j times k in j.
    Store the value of k times l in l.
    Add j and l, and store the result in k.
    Display the value in k on the screen.

35. The variable a starts with the value 1.
    The variable b starts with the value 10.
    The variable c starts with the value 100.
    The variable x starts with the value 0.
    Store the value of c times 3 in x.
    Add the value of b times 6 to the value already in x.
    Add the value of a times 5 to the value already in x.
    Display the value in x on the screen.

p28

Program 2-1

```
// A simple C++ program
#include <iostream>
using namespace std;

int main()
{
   cout << "Programming is great fun!";
   return 0;
}
```

The output of the program is shown below. This is what appears on the screen when the program runs.

Program Output

```
Programming is great fun!
```

Figure 1-9 CCC Monday 1/11/2016

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools such as a hierarchy chart, flowcharts, or pseudocode to create a model of the program.
4. Check the model for logical errors.
5. Type the code, save it, and compile it.
6. Correct any errors found during compilation. Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any errors found while running the program. Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

Program 2-1

```
// A simple C++ program
#include <iostream>
using namespace std;

int main()
{
   cout << "Programming is great fun!";
   return 0;
}
```

The output of the program is shown below.

Program Output

```
Programming is great fun!
```