# **Design and Verify Program Algorithm**

* **Design program algorithm using IPO chart**

**Example 1**: *The NC Speedy Bike Rental Manager has asked you to create a program that calculates the amount a customer owes when a bike is returned. Assume that a customer can return only one bike at a time and he/she pays the rental fee and any late fee when the bike is returned. The rental fee is $50 for per day. Customers are charged a late fee of $100.00 per day when the bike is returned past the due date.*

Let’s make an **IPO (Input-Process-Output) chart** for this problem. An IPO chart consists of three columns: **Input**, **Processing**, and **Output**. One good way to make an IPO chart is to start with output. In this example, the goal of the program is to determine the amount a customer owes. Therefore, the amount due is inserted in the output column.

| Input | Processing (Algorithm) | Output |
| --- | --- | --- |
|  |  | amount due |

Next, let’s think about what information the user of the program needs to enter to the program in order to get the desired output. The rental fee and the late fee per day are fixed at $50.00 and $$100, respectively. To save time, the manager does not want the store associate to enter these two numbers every time this program is used. Therefore, we are going to put these numbers directly in the program calculation code. The disadvantage of this approach is that if there is a change in the rental fee or late fee, the program must be modified. Since these fees do not change frequently, fixing them to $50.00 and $$100 inside the program should not be a problem. The only thing that changes from one customer to another is how many days the bike is late. To determine how much money a customer owes, the user of the program must enter the number of late days when the bike is returned.

| Input | Processing | Output |
| --- | --- | --- |
| number of days overdue |  | amount due |

Now the input and output items of the program are clear. The next question is how the program yields the desired output. In this example, we need to think about the steps the program takes to determine the total amount due from the customer. First, the program must know how many days the bike is overdue. How does the program know that? Well, it asks the user to enter this information into the program. This is the first step.

Next, the program calculates the total late fee by multiplying the number of days overdue by the late fee per day, which is $100. This is step 2.

In step 3, the program calculates the total amount due from adding the total late fee to the rental fee, which is $50.00.

Now the program knows the total amount due. It has to display it on the computer screen so that the user of the program can see it. This is step 4.

These steps together form the algorithm. In IPO chart, we put the algorithm in the Processing column.

| Input | Processing (Algorithm) | Output |
| --- | --- | --- |
| number of days overdue | Algorithm:  1. enter the number of days overdue  2. calculate the total late fee by multiplying the number of days overdue by $100  3. calculate the amount due by adding the total late fee to 50.00(i.e. the rental fee)  4. display the amount due | amount due |

Let’s look at the nature of each step in the algorithm. Step 1 is an input step. Its purpose is to get the necessary input from the user of the program. Step 2 and Step 3 are calculation steps. The goal here is to yield the desired output by performing some calculations with the user input and other information. Step 4 is an output step. The user will never get the output if it is not displayed by the program. This model of “get input, then calculate, and finally display output” applies to many programs we are going to write in this semester.

Some people may like to use equations instead of text to express calculation steps. Step 2 and Step 3 can be rewritten like this:

| Input | Processing (Algorithm) | Output |
| --- | --- | --- |
| number of days overdue | Algorithm:  1. enter the number of days overdue  2. total late fee = number of days overdue \* $100  3. amount due = total late fee + 50.00  4. display the amount due | amount due |

Although there is no universal standard on how to write the steps in an algorithm, let’s set up a standard ourselves:

1. For steps about getting input from the user, let’s use this format:

**enter *[list of input items]***

Example: enter number of days overdue

2. For steps about calculations, use this format:

**calculate *[result by a description of how to calculate the result]***

Example: calculate total late fee by multiplying the number of days overdue by $100

If you like to use equation, use this format:

***[result]* = *[mathematical expression that yields the result]***

Example: total late fee = number of days overdue \* $100

3. For steps about displaying output to the user, let’s use this format:

**display *[list of output items]***

Example: display the amount due

Please follow this standard when you make IPO charts in this course.

We need one more thing to complete the IPO chart. In the Processing column, in addition to describing the steps in the algorithm, we also need to list the processing items.

What is a processing item? A processing item is an item the program calculates only for the purpose of using it later in another calculation step. A processing item is not an output item because the user is not interested in this item itself. The program will never display a processing item to the user. If an item is displayed by the program, then it is an output item, not a processing item. A processing item is not an input item because a user never enters this item directly to the program. If an item is entered into the program by the user, then that item is an input item, not a processing item. A processing item must be an item generated in a calculation step. It exists in a program because the program cannot find the output without calculating the processing item first.

If you examine the calculation steps in the example, you will find that two items are calculated. In Step 2, the program calculates the total late fee from the number of days overdue and the late fee per day. In step 3, the program calculates the total amount due from the total late fee and the rental fee. The total amount due is an output item because the user of the program is interested to see how much money the customer owes. The total late fee, which is calculated in Step 2, is a processing item because the user is not interested in knowing how much late fee the customer owes (unless the question says that the user wants to see this item). In fact, the only reason the total late fee is calculated is that the program needs this item when it calculates the total amount due in Step 3. This makes the total late fee a processing item, not an output item.

In an IPO chart, we list processing items in the Processing column.

| Input | Processing (Algorithm) | Output |
| --- | --- | --- |
| number of late days | Processing items: total late fee  Algorithm:  1. enter the number of late days  2. total late fee = number of late days \* $100  3. amount due = total late fee + 50.00  4. display the amount due | amount due |

The chart above is a complete IPO chart for this example. It presents a solution to a problem, including the steps it takes to achieve the desired goal, and the list of items involved.

The simple statements/equations used to describe the algorithm in the IPO chart above is called **pseudocode**. Some people like to use a more graphical representation, i.e. a flowchart, to show the steps in an algorithm

* **Test and verify the program algorithm (Desk-Checking / Hand-Tracing):**

Next, let’s make up the first test case. Assuming that one customer returns a bike 2 days late. Let’s walk through the steps in the algorithm to see whether the correct output is generated.

Step 1 in the algorithm is to get the number of days overdue from the user. Suppose the user enters 2 to the program. At this point the program has the following information:

| number of late days | total late fee | amount due |
| --- | --- | --- |
| 2 |  |  |

Next, the program executes Step 2 to calculate total late fee, which is the product of late fee per day and the number of days overdue. In this case, the total late fee is $100 times 2, which is $200.00. Let’s enter this into the desk checking table.

| number of late days | total late fee | amount due |
| --- | --- | --- |
| 2 | 200.00 |  |

In Step 3, the amount due is calculated by finding the sum of total late fee and rental fee. In this case, it is $200.00 plus $50.00, which is $7.50.

| number of late days | total late fee | amount due |
| --- | --- | --- |
| 2 | 200.00 | 250.00 |

This finishes the first **test case**. We encountered no trouble in finding the output.

Let’s try another test case. This time assume that a customer returns a bike 5 days late. Let’s see how the **desk checking table** looks like when the second test case is added there.

| number of late days | total late fee | amount due |
| --- | --- | --- |
| 2  5 | 200.00  500.00 | 250.00  550.00 |

Again, we encountered no trouble in getting the output.

* **Reading Data from a Sequential Access File**

The following is an example of inputting data from a sequential access file.

***Example 2:*** *Data from 3 participants in a fitness training class were collected and stored in the file “fitness.txt”. In the file, data for each participant are stored on a separate line in this order: id, age, weight, height, systolic pressure and diastolic pressure. The following program reads the data form the file. It also calculates and displays BMI for each participant.*

A sequential access file is a file in which data are stored and retrieved sequentially in consecutive order. You can use text editor such as Notepad to view or even change its content. For example, the file “fitness.txt” may contain the following data:

1263 32 148 69 132 87

3874 28 121 65 120 81

9044 51 132 62 130 98

IPO chart:

| Input | Processing | Output |
| --- | --- | --- |
| id  age  weight  height  systolic pressure  diastolic pressure | Processing items: counter  Algorithm:  1. Open input file “fitness.txt” for input  2. Read id, age, weight, height, systolic and diastolic pressure from input file.   1. Calculate BMI = 703 \* weight / (height \* height) 2. Display id, BMI 3. Close input file | id  BMI |

Desk checking table:

| id | age | weight | height | systolic pressure | diastolic pressure | BMI |
| --- | --- | --- | --- | --- | --- | --- |
| 1263  3874  9044 | 32  28  51 | 148  121  132 | 69  65  62 | 132  120  130 | 87  81  98 | 21.8534  20.1333  24.1405 |