# Translating IPO Chart into C++ Code

We’ve learned how to analyze a problem and create an IPO chart to summarize the solution from the Algorithm handout. The next step is to translate the IPO chart into C++ statements. This includes: (a) assign a name, data type and initial value to every unique input, processing and output item; (b) translate the algorithm steps into C++ statements.

* **Assign a name, data type and initial value to every unique input, processing and output item**

Each input, processing and output item on the IPO chart should be translated into a ***variable*** in a C++ program. Consider a variable as a memory location with a name and a type associated to it. The type of a variable determines what kind of information can be stored in the memory location. The types we use most in this course are as follows:

***int***: for memory locations that store integer values (e.g. 7, -15, etc.)

***double***: for memory locations that store decimal values (e.g. 4.7, -2.0, etc.)

***char***: for memory locations that store characters (e.g. ‘K’, ‘a’, ‘:’, etc.)

The initial value of a variable refers to what is stored in the memory location right after the variable is created. Although it is not required, it is a good programming habit to explicitly give each variable an initial value when a variable is created. Typically, if the type of the variable is *int*, we assign an initial value of 0 to the variable. If the type of the variable is *double*, we assign an initial value of 0.0. If the type of the variable is *char*, we assign an initial value of ' ' to the variable (Note: There is a blank space between the quotes).

There are some rules on how to name a variable.

1. The name must begin with a letter.

2. The name must contain only letters, numbers, and the underscore. No punctuation characters or spaces are allowed in the name.

3. The C++ compiler you are using determines the maximum number of characters in a name. Although names can contain thousands of characters, the recommended maximum number of characters to use is 32.

4. The name cannot be a keyword, such as double, because a keyword has a special meaning in C

5. Names in C++ are case sensitive.

The following are examples of how to make up names and assign types and initial values to variables:

int numPets = 0;

double gpa = 0.0;

char letterGrade = ' ';

* **Translate the algorithm steps into C++ statements**

In the programs we are going to write in this and the next labs, we will use only three types of statements: input statements, output statements and assignment statements.

## **Input statements**

To get input into a C++ program, we use a *cin* statement, which has the following syntax:

cin >> variable\_name;

The word *cin* tells the computer to read what the user enters through the keyboard and store the input value in the variable variable\_name. Example:

cin >> numPets;

Suppose the user types 2, the computer will read the 2 and store it in the variable numPets.

## **Output statements**

To display information on the computer screen, we use a *cout* statement, which has the following syntax:

cout << output\_item;

There are two types of items we can display on the screen: a text message or the value stored in a variable. To display a text message, we put the message inside double quotes. Example:

cout << "I have 2 pets.";

To display the value stored in a variable on the screen, we simply put the name of the variable in the *cout* statement. Example:

cout << numPets;

Suppose 2 is stored in the variable numPets, we will see 2 on the screen. In this example, we may want to also display a text message on the screen to clarify what the 2 means:

cout << "Number of pets I have: ";

cout << numPets;

This time we will see this output on the screen:

Number of pets I have: 2

We can combine the two *cout* statements into one:

cout << "Number of pets I have: " << numPets;

We can also do this:

cout << " I have " << numPets << " pets.";

Suppose 2 is stored in the variable numPets, what will you see on the screen?

There is one more thing about the *cout* statement. The computer does not automatically move the cursor to the next line on the screen when a *cout* statement ends. Look at this example:

cout << "I have two pets.";

cout << "One of them is a cat while the other one is a dog.";

This is what we will see on the screen:

I have two pets. One of them is a cat while the other one is a dog.

We can use the word *endl* to move the cursor to the next line. Example:

cout << "I have two pets." << endl;

cout << "One of them is a cat while the other one is a dog.";

This is what we will see on the screen now:

I have two pets.

One of them is a cat while the other one is a dog.

## **Assignment statements**

We use an assignment statement to assign a value to a variable. The syntax:

Variable\_name = value\_or\_mathematical\_expression;

Examples:

taxRate = 0.07;

price = price\_before\_tax + price\_before\_tax \* taxRate;

If the right hand side of an assignment statement is a value, the value will be assigned to the variable on the left hand side. If the right hand side is a mathematical expression, the computer will first evaluate the expression (i.e. calculate the value) and then assign the value to the variable on the left hand side.

* **Examples**

The following examples show us how to translate items on IPO chart into C++ code.

**Example 1**

An online music store sells CDs for $8.99 each. Shipping charge is $2.99 per order (not per CD). Write a program that asks for the number of CDs purchased and then calculates and displays the total amount due on the screen.

First, let’s see the IPO chart and the flowchart:

| Input | Processing | Output |
| --- | --- | --- |
| number of CDs | Processing items: none  Algorithm:  1. enter the number of CDs  2. total amount due = number of CDs \* 8.99 + 2.99  3. display total amount due | total amount due |

Let’s make a table to translate the items on IPO chart to C++ instructions:

| IPO chart information | C++ instructions |
| --- | --- |
| Input  number of CDs    Processing  Output  total amount due  Algorithm  enter number of CD purchased  total amount due = number of CDs\*8.99 + 2.99  display total amount due | int numberCD = 0;  double totalDue = 0.0;  cout << "Enter number of CDs purchased: ";  cin >> numberCD;  totalDue = numberCD \* 8.99 + 2.99;  cout << "Total amount due: " << totalDue; |

**Example 2**

A restaurant needs a program to print checks. The program asks for the total amount for food, calculate and display the sales tax and the total amount due (including tax). The tax rate is 7%.

| Input | Processing | Output |
| --- | --- | --- |
| total amount for food | Processing items: none  Algorithm:  1. enter total amount for food  2. sales tax = total amount for food \* 0.07  3. total amount due = total amount for food + sales tax  4. display sales tax and total amount due | sales tax  total amount due |

Let’s make a table to translate the items on IPO chart to C++ instructions:

| IPO chart information | C++ instructions |
| --- | --- |
| Input  total amount for food    Processing    Output  sales tax  total amount due  Algorithm  enter total amount for food.  sales tax = total amount for food \* 0.07  total amount due = total amount for food + sales tax    display sales tax and total amount due. | double totFood = 0.0;  double tax = 0.0;  double totalDue = 0.0;  cout << "Enter total for food: ";  cin >> totFood;  tax = totFood \* 0.07;  totalDue = totFood + tax;  cout << "Sales Tax: " << tax << endl;  cout << "Total amount due: " << totalDue << endl; |

**Example 3**

A teacher needs a program to calculate a student’s total score in a course. Each student needs to take a mid-term exam and a final exam. Scores are in the range of 0 to 100. The mid-term counts as 40% while the final counts as 60% of the total score. The program should ask for the exam scores. It should calculate and display the total score on the screen.

| Input | Processing | Output |
| --- | --- | --- |
| midterm score  final score | Processing items: none  Algorithm:  1. enter midterm score and final score  2. total score = midterm score \* 0.4 + final score \* 0.6  3. display total score | total score |

Translation of items on IPO chart to C++ instructions:

| IPO chart information | C++ instructions |
| --- | --- |
| Input  midterm score  final score    Processing  Output  total score  Algorithm  enter midterm score and final score.  total score = midterm score \* 0.4 + final score \* 0.6  display total score. | int midterm = 0;  int final = 0;  double totalScore = 0.0;  cout << "Enter mid-term exam score: ";  cin >> midterm;  cout << "Enter final exam score: ";  cin >> final;  totalScore = midTerm \* 0.4 + final \* 0.6;  cout << "Total score: " << totalScore << endl; |