The Program Development Process

1. Problem Analysis
   - Input - *What is given?*
   - Output - *What is required?*
   - Constraints - *Under what conditions?*
   - Abstraction - *What information is essential?*
2. Program Design
   - Decomposition
   - Initial Algorithm Development
   - Algorithm Refinement
3. Program Implementation
   - Coding - Translate the algorithm into program using a programming language.
4. Program Debugging & Testing
5. Program Documentation

What Constitute a good program?

1. Correct
   - Meet the problem requirements
   - Produce correct results
2. Easy to read and understand
3. Easy to modify
4. Easy to debug and maintain
5. Efficient
   - Fast
   - Requires less memory

Write an algorithm for a robot to swap two pictures on the wall of a room.

1. Position yourself in front of one of the two pictures to be swapped. (Call the location L1 and the picture P1.)
2. Take picture P1 off the wall.
3. Carry picture P1 to location (L2) of other picture (P2) to be swapped.
4. Set picture P1 down safely on floor.
5. Remove picture P2 from wall.
6. Take picture P2 to location L1.
7. Hang picture P2 on the wall.
8. Go to location L2.
9. Pick up picture P1.
10. Hang picture P1.

(Notice that there are two trips being made to L2. Solve the same problem without the extra trip. This would be a more "efficient" solution.)

Write an algorithm for a robot to obtain a soft drink from a vending machine. Assume soft drinks cost $5 apiece and the robot has $2 coins only.

1. Go to softdrink machine.
2. Check if "out of change" light is on.
3. Insert coins.
4. Press drink selection button.
5. Check if the drink is vended.

A refined solution:

1. Go to softdrink machine.
2. Check if "out of change light is on:
   - 2.1. If "out of change" light is on,
     obtain correct change or quit.
3. Insert coins:
   - 3.1. Insert first $2 in coin slot.
   - 3.2. Insert second $2 in coin slot.
   - 3.3. Insert third $2 in coin slot.
4. Press drink selection button.
5. Check if the drink is vended:
   - 5.1. If the drink is vended,
     - 5.1.1. Remove drink from machine.
     - 5.1.2. Remove change from change hopper.
   - 5.2. If drink is not vended,
     - 5.2.1. Pound on machine.
     - 5.2.2. Press drink selection button repeatedly.
     - 5.2.3. If still not vended, press return lever.
     - 5.2.4. If change is returned, try again or go away.
     - 5.2.5. If change is not returned, go away.

Top-down Design and Modularisation

Motivation

- Large problem are difficult to solve as a whole.

Modularisation

- ***Dividing*** a problem into subproblems (or modules)
- It is usually done in a ***top-down*** manner.
- The problem is solved by solving the subproblems.
- If a subproblem is still difficult, (***recursively***) dividing it further into smaller subproblems.
- A module should be small enough to be handled in one piece and large enough to be considered as a task.

Advantage of Modularisation

- makes programming simpler, more systematic, and more likely to be free of errors.
- Each module can be clearly specified, designed, refined, developed, and tested individually.
- Some modules can be **reused.**

Problem Statement

You are given a collection of nickels (US 5 cents) and pennies (US 1 cent) coins, find the number of Hong Kong dollars and 10-cent coins in exchange.

Problem Analysis

Input:

- nickels (integer) - count of nickels
- pennies (integer) - count of pennies

Output:

- dollars (integer) - count of HK dollar in return
- change (integer) - count of HK 10-cent coins in return

Constraints: None

Program Design

Knowledge:

- one dollar equals 100 cents

Problem Decomposition:

- Find the total amount in US dollars

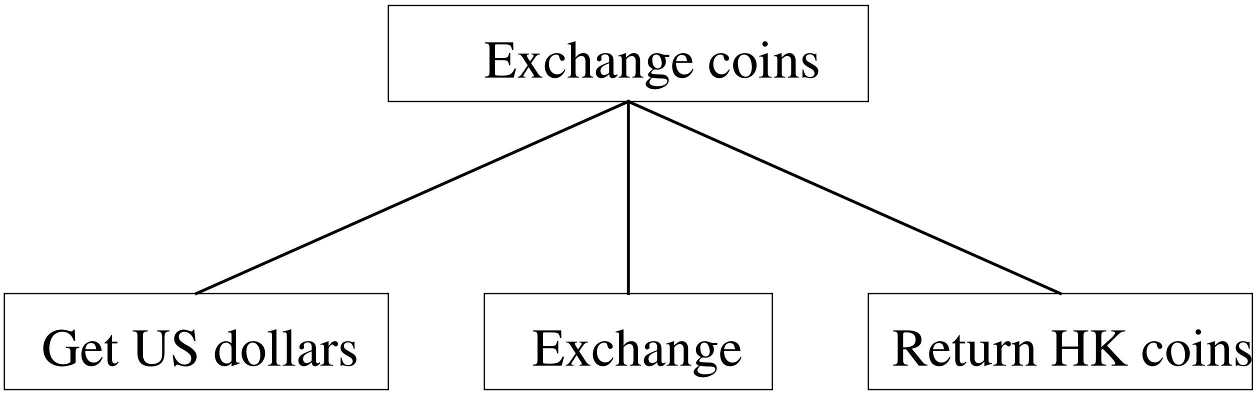
  - Input: nickels and pennies
  - Output: total_US_dollars

- Find the total amount of HK dollars in exchange

  - Input: total_US_dollars
  - Output: total_HK_dollars

Find the number of HK dollars and 10-Cent coins

- Input: total_HK_dollars
- Output: dollars and change



New variables:

- total_US-dollars (real) - total amount of US dollars
- total_HK_dollars (real) - total amount of HK dollars

Initial Algorithm:

1. Read in the count of nickels and pennies
2. Compute the total amount in US dollars
3. Compute the total amount in HK dollars in exchange
4. Find the value in HK dollars and change
5. Display the value in HK dollars and change

Refinement 1:

1. Read in the count of nickels and pennies
2. Compute the total amount in US dollars

   2.1. total_US_dollars = 5 * nickels + pennies

3. Compute the total amount in HK dollars in exchange

   3.1. total_HK_dollars = total_US_dollars * US2HK

4. Find the value in HK dollars and change

   4.1. total_HK_cents = total_HK_dollars * DOLLAR2CENTS
   4.2. dollars = total_HK_cents / DOLLAR2CENTS
   4.3. change = total_HK_cents%DOLLAR2CENTS/10

5. Display the value in HK dollars and change

The C++ main program skeleton:

```
// File :  excoin.cpp
// Determines the number of HK coins in exchange of US coins

#include <iostream.h>

void main ()
{

        int nickels;                    // count of nickels
        int pennies;                    // count of pennies
        int dollars;                    // value of coins in HK dollars
        int change;                     // value of coins in HK 10-cents
        float total_US_dollars;         // total US dollars
        float total_HK_dollars;         // total HK dollars


        // Read in the count of nickels and pennies

        // Compute the total amount in US dollars

        // Compute the total amount in HK dollars in exchange

        // Find the value in HK dollars and change

        // Display the value in HK dollars and change
}
```

```
// File :  excoin.cpp
// Determines the number of HK coins in exchange of US coins

#include <iostream.h>

void main ()
{
	// 1 dollar = 100 cents
	const int DOLLAR2CENTS = 100;
	// assume exchange rate is US$1 = HK$7.8
	const float US2HK = 7.8;

	int nickels;				// count of nickels
	int pennies;				// count of pennies
	int dollars;				// value of coins in HK dollars
	int change;				// value of coins in HK 10-cents
	int total_US_cents;			// total cents in US
	float total_US_dollars;		// total US dollars
	int total_HK_cents;			// total cents in HK
	float total_HK_dollars;		// total HK dollars

	// Read in the count of nickels and pennies
	cout << "Enter the number of nickels and press return: ";
	cin >> nickels;
	cout << "Enter the number of pennies and press return: ";
	cin >> pennies;

	// Compute the total amount in US dollars
	total_US_cents = 5 * nickels + pennies;
	total_US_dollars = (float) total_US_cents / DOLLAR2CENTS;

	// Compute the total amount in HK dollars using the exchange rate
	total_HK_dollars = total_US_dollars * US2HK;

	// Find the value in HK dollars and change
	total_HK_cents = (int) (total_HK_dollars * DOLLAR2CENTS);
	dollars = total_HK_cents / DOLLAR2CENTS;
	change = total_HK_cents % DOLLAR2CENTS / 10;

	// Display the value in HK dollars and change
	cout << "The change is HK " << dollars << " dollars and "
		<< change << " 10-cents." << endl;
}
```