บทที่ 1

ความรู้เบื้องต้นเกี่ยวกับโปรแกรมคอมพิวเตอร์สำเร็จรูปทางคณิตศาสตร์

1.1 โปรแกรมคอมพิวเตอร์ขั้นแนะนำ

บทนิยาม 1.1.1

โปรแกรมคอมพิวเตอร์ (computer program) (หรือเรียกสั้น ๆ ว่า **โปรแกรม** ก็ได้) คือ ชุดของคำสั่งคอมพิวเตอร์ที่มนุษย์สร้างขึ้น เพื่อให้คอมพิวเตอร์ทำงานอย่างใดอย่างหนึ่งได้

ตัวอย่าง 1.1.2 พิจารณาความแตกต่างระหว่างโปรแกรม (program) และคำสั่ง (command) ในสถานการณ์ต่อไปนี้

คำสั่ง		- เลี้ยวขวา (... เมื่อเลี้ยวขวาแล้ว ถือว่าทำงานเสร็จ เว้นแต่จะสั่งคำสั่งใหม่ต่อไป ...)
โปรแกรม	 ราคารวมแล้วไม่เกินเงินที่เหลืออยู่ - นำเงินที่เหลืออยู่ (ถ้ามี) คืนภรรยา	- รับเงินจากภรรยามา 500 บาท - ถ้าน้ำมันรถเหลืออยู่ไม่เกิน 2 ลิตร ให้เติมน้ำมันได้ 100 บาท - เลี้ยวขวา - ตรงไปอีก 1 กิโลเมตร จะถึงตลาด - ให้ซื้อขนุนมาเป็นจำนวนกิโลกรัมเต็ม โดยต้องมี

โดยทั่วไปแล้ว โปรแกรมคอมพิวเตอร์จะมีองค์ประกอบที่สำคัญดังต่อไปนี้

1. **สิ่งเข้า** (input) คือ ข้อมูลตั้งต้นที่โปรแกรมต้องการทราบ เพื่อใช้ในการคำนวณต่อไป เช่น ถ้าจะเขียนโปรแกรมเพื่อคำนวณหาเกรดเฉลี่ย เราต้องทราบเกรดและหน่วยกิตของแต่ละวิชาเสียก่อน ดังนั้นเกรดและหน่วยกิตของแต่ละรายวิชาจึงเป็น input ของโปรแกรมดังกล่าว เป็นต้น
2. **สิ่งออก** (output) คือ ข้อมูลสุดท้ายที่โปรแกรมสร้างขึ้น แล้วส่งออกไปยังผู้ใช้โปรแกรม เช่น ในโปรแกรมหาเกรดเฉลี่ยข้างต้น output ได้แก่เกรดเฉลี่ย ซึ่งเป็นจำนวนจริงที่มีทศนิยม 2 ตำแหน่ง เป็นต้น
3. **ส่วนประมวลผล** (processing component) คือ ส่วนที่ประกอบด้วยคำสั่งต่าง ๆ ที่จำเป็นในคำนวณ เปรียบเสมือนเป็น “สมอง” ของโปรแกรม ส่วนประมวลผลอาจแบ่งออกได้เป็นองค์ประกอบย่อย ๆ ได้อีก ดังนี้
   (ก) **ชุดคำสั่งที่ช่วยในการดำเนินการเลขคณิต** (arithmetic-operation commands) เช่น ชุดคำสั่งที่ทำหน้าที่บวก ลบ คูณ หารจำนวนจริงสองจำนวน เป็นต้น

(ข) **ชุดคำสั่งที่ช่วยในการตัดสินใจ** (decision-making commands) เป็นส่วนที่กำหนดทิศทางของโปรแกรมโดยตรวจสอบกับเงื่อนไขบางอย่าง เช่น ตรวจสอบดูว่าตัวแปร x มีค่าเป็นบวกหรือไม่ ถ้าเป็นก็ให้หา x^2 แต่ถ้าไม่เป็นก็ให้หา $|x|$ แทน เป็นต้น

(ค) **ชุดคำสั่งที่ช่วยในการทำซ้ำ** (iteration commands) เป็นส่วนที่ดำเนินการทำตามชุดคำสั่งหนึ่ง ๆ ซ้ำไปเรื่อย ๆ จนกว่าจะได้จำนวนรอบตามที่กำหนด หรือจนกว่าเงื่อนไขบางอย่างจะเป็นจริง เช่น ให้คำนวณหา $\sqrt{n}$ เมื่อ $n = 1, 2, 3, \ldots, 10$ เป็นต้น

4. **ส่วนแสดงผล** (display component) เป็นส่วนที่ประกอบไปด้วยคำสั่งต่าง ๆ ที่ทำการติดต่ออุปกรณ์ที่ใช้ในการแสดงผล เช่น จอภาพ (monitor) ลำโพง (speaker) เครื่องพิมพ์ (printer) เป็นต้น ตัวอย่างเช่น ในโปรแกรมหาเกรดเฉลี่ย เราอาจต้องการให้โปรแกรมพิมพ์ผลออกทั้งทางจอภาพและเครื่องพิมพ์ด้วย เป็นต้น

ข้อสังเกต โปรแกรมคอมพิวเตอร์บางโปรแกรมอาจขาดองค์ประกอบใดองค์ประกอบหนึ่งข้างต้นก็ได้

ตัวอย่าง 1.1.3 จงวิเคราะห์องค์ประกอบในเชิงโปรแกรมคอมพิวเตอร์ของเครื่องซักผ้า

วิธีทำ

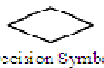
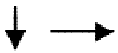
ในการออกแบบโปรแกรม (program designing) นั้น เราจะต้องพิจารณาคำถามต่อไปนี้เสียก่อน

1. ต้องการให้โปรแกรมทำอะไร (นั่นคือ ต้องการให้ output ของโปรแกรมออกมาเป็นอะไร)
2. เพื่อให้ได้ output ข้างต้น ต้องใช้ input อะไรบ้าง
3. เมื่อมี input แล้ว ต้องดำเนินการอย่างไรจึงจะได้ output ตามที่ต้องการ (นั่นคือ ต้องคิดว่าควรทำอะไรเป็นอันดับแรก อันดับสอง และอันดับต่อ ๆ ไป)

การออกแบบโปรแกรมคอมพิวเตอร์นั้น เป็นคนละขั้นตอนกับการเขียนโปรแกรมให้อยู่ในรูปของภาษาคอมพิวเตอร์ (program implementation) หากแต่เป็นเพียงการลำดับความคิดของเราออกมาให้เป็นประโยคที่เราเข้าใจได้ง่าย เราเรียกประโยคเหล่านี้ว่า **รหัสเทียม** (pseudocode)

อย่างไรก็ตาม การเขียนรหัสเทียมบ่อยครั้งมักไม่เป็นการสะดวกนัก จึงได้มีการพัฒนาการเขียนโปรแกรมให้อยู่ในรูปของ **ผังงาน** (flowchart) โดยใช้สัญลักษณ์แทนองค์ประกอบต่าง ๆ ของโปรแกรม ซึ่งโดยทั่วไปแล้วมีลักษณะดังต่อไปนี้

ตารางที่ 1.1 : สัญลักษณ์ที่ใช้ในการเขียนผังงาน

ภาพสัญลักษณ์	ความหมาย
 Start/End Symbol	เริ่มต้น/สิ้นสุด, การเริ่มต้นหรือการลงท้าย
 Connection Symbol	จุดเชื่อมต่อในหน้าเดียวกัน
 Connection Symbol	จุดเชื่อมต่อคนละหน้า
 Monitor	จอภาพแสดงผล
 Processing	การประมวลผลทั่วไป ยกเว้นการอ่านข้อมูลและการแสดงผลลัพธ์
 Input/Output Data	รับหรือแสดงข้อมูล โดยไม่ระบุชนิดหรืออุปกรณ์
 Decision Symbol	การตัดสินใจ การเปรียบเทียบ (จะมีทิศทางออก 2 ทิศทาง คือกรณีที่ผลตรวจสอบเงื่อนไขเป็นเท็จและเป็นจริง)
 Manual input	การรับข้อมูลเข้าทางแป้นพิมพ์
 Document Output	เอกสารแสดงผล, การแสดงผลทางเครื่องพิมพ์
 Preparation	ใช้กำหนดค่าต่างๆส่วงหน้า ซึ่งเป็นการทำงาน ภายในช่วงหนึ่งที่ซ้ำๆกัน
 Flow line	เส้นแสดงลำดับกิจกรรม

(**ที่มา :** http://สัญ-ลักษณ์.blogspot.com/2013/03/flowchart.html)

เนื่องจากสัญลักษณ์เหล่านี้มักเป็นที่นิยมใช้และเป็นที่เข้าใจตรงกันในวงการคอมพิวเตอร์ ทำให้โปรแกรมคอมพิวเตอร์ที่แสดงในรูปของผังงานนั้นมีความหมายที่แน่นอนและดูเป็นสากลกว่า เนื่องจากไม่ผูกมัดกับภาษาใด ๆ จึงสามารถใช้สื่อความหมายระหว่างนักเขียนโปรแกรม (programmer) ด้วยกันได้ดี

ตัวอย่าง 1.1.4 จงออกแบบโปรแกรมที่คำนวณหาค่าของฟังก์ชัน $f : \mathbb{N} \to \mathbb{R}$ ที่นิยามโดย

$$f(n) = \begin{cases} 2 - n & \text{ถ้า } 1 \leq n < 4 \\ \dfrac{n}{2} & \text{ถ้า } 4 \leq n < 8 \\ n^2 + 3n & \text{ถ้า } n \geq 8 \end{cases}$$

ให้อยู่ในรูปของรหัสเทียมและผังงาน ตามลำดับ

<u>วิธีทำ</u>

1.2 โปรแกรมคอมพิวเตอร์สำเร็จรูปทางคณิตศาสตร์

ในการคำนวณทางคณิตศาสตร์นั้น มักจะเกี่ยวข้องกับฟังก์ชัน (functions) ตัวดำเนินการ (operators) และโครงสร้างทางคณิตศาสตร์ (mathematical structures) เป็นจำนวนมาก การที่เราจะเขียนโปรแกรมเพื่อทำงานหนึ่ง ๆ ให้สำเร็จนั้น โดยหลักแล้วจะต้องมีการเขียนโปรแกรมเพื่อสร้างฟังก์ชัน ตัวดำเนินการ และโครงสร้างทางคณิตศาสตร์ที่จำเป็นเสียก่อนเสมอ ซึ่งเป็นการไม่สะดวกอย่างมากหากภาษาคอมพิวเตอร์ที่เราต้องใช้นั้นไม่มีสิ่งเหล่านี้แต่แรก เช่น ถ้าเราต้องการเขียนโปรแกรมเพื่อหาผลบวกของจำนวนเชิงซ้อน (complex numbers) 2 จำนวนโดยใช้ภาษาซี (C language) เราจะต้องกำหนดโครงสร้างของจำนวนเชิงซ้อน และฟังก์ชันที่จะหาผลบวกของจำนวนเชิงซ้อน 2 จำนวนเอง เนื่องจากภาษาซีไม่มีสิ่งเหล่านี้ตั้งแต่แรก เป็นต้น (ดูตัวอย่าง 1.2.1)

จากปัญหาดังกล่าว จึงได้มีผู้พัฒนา **โปรแกรมคอมพิวเตอร์สำเร็จรูปทางคณิตศาสตร์** (mathematical computer packages) ขึ้นเป็นจำนวนมาก ทั้งแบบเพื่อจำหน่ายในเชิงพาณิชย์ และแบบที่สามารถนำไปใช้ได้โดยไม่มีค่าใช้จ่าย เพื่อช่วยอำนวยความสะดวกให้นักเขียนโปรแกรมสามารถเขียนโปรแกรมสำหรับงานที่ตนต้องการได้รวดเร็วขึ้น โดยไม่จำเป็นต้องทราบรายละเอียดในเชิงลึกของฟังก์ชัน ตัวดำเนินการ และโครงสร้างทางคณิตศาสตร์ที่อยู่เบื้องหลังมากนัก

ตัวอย่าง 1.2.1 พิจารณาการเขียนโปรแกรมเพื่อหาผลบวกของจำนวนเชิงซ้อน $a = 2 + 3i$ และ $b = 1 - i$ (เมื่อ $i^2 = -1$) โดยใช้ภาษาซี เปรียบเทียบกับการใช้โปรแกรมคอมพิวเตอร์สำเร็จรูปทางคณิตศาสตร์ (ในที่นี้ใช้โปรแกรม Scilab ซึ่งจะกล่าวถึงในหัวข้อต่อไป)

ภาษาซี

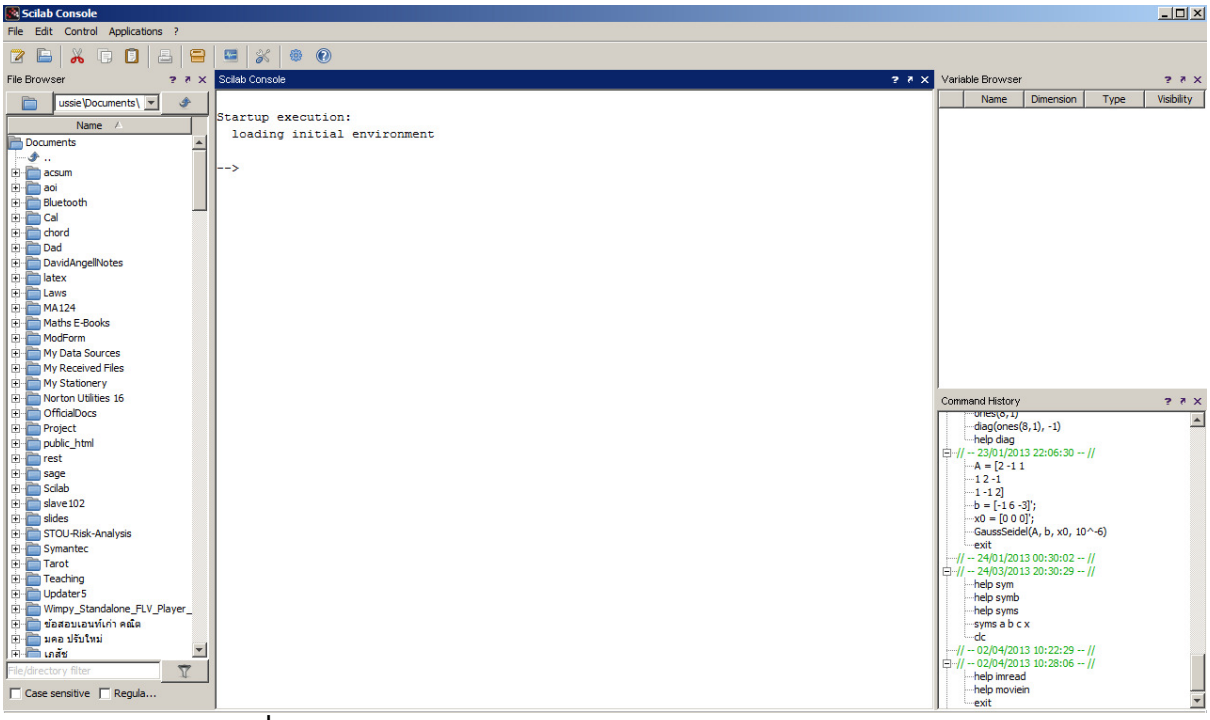
```
#include <stdio.h>
typedef struct complex complex;

// Define the structure of complex numbers
struct complex {
	float x; // real part
	float y; // imaginary part
};

// Function for adding two complex numbers
complex add(complex a, complex b){
	// a = x1 + y1*i
	float x1 = a.x;
	float x2 = b.x;
	// b = x2 + y2*i
	float y1 = a.y;
	float y2 = b.y;
	complex c = {x1+x2, y1+y2}; // c = a + b = (x1+x2) + (y1+y2)*i
	return c;
}

// Main program
int main(void){
	complex a = {2, 3}; // Define a = 2+3i
	complex b = {1, -1}; // Define b = 1-i
	complex c = add(a, b); // c = a+b
	printf("c = %f+%f i\n", c.x, c.y);
	return 0;
}
```

ภาษาสำหรับโปรแกรม Scilab

```
i = sqrt(-1); // Define i
a = 2 + 3*i;
b = 1 - i;
c = a+b
```

1.3 การใช้งานโปรแกรม Scilab ขั้นแนะนำ

Scilab เป็นโปรแกรมคอมพิวเตอร์สำเร็จรูปทางคณิตศาสตร์ (mathematical programming package) โปรแกรมหนึ่งซึ่งพัฒนาขึ้นโดย Institut national de recherche en informatique et en automatique (INRIA) ซึ่งเป็นสถาบันวิจัยแห่งชาติทางด้านวิทยาการคอมพิวเตอร์ของประเทศฝรั่งเศส Scilab เป็นโปรแกรมที่เหมาะสำหรับการคำนวณเชิงตัวเลข (numerical computation) ใช้งานได้ง่าย และสามารถใช้งานได้โดยไม่เสียค่าใช้จ่าย (download โปรแกรมและคู่มือการใช้งานได้ที่ http://www.scilab.org)



รูปที่ 1.1 : หน้าต่างหลัก (main window) ของโปรแกรม Scilab

เมื่อเข้าสู่หน้าต่างหลักของโปรแกรม Scilab แล้ว เราจะพบกับหน้าต่างย่อย 4 หน้าต่าง ได้แก่

1. **File Browser** ใช้สำหรับค้นหาไฟล์ในโฟลเดอร์ต่าง ๆ
2. **Scilab Console** เป็นหน้าต่างที่ใช้รับคำสั่งในลักษณะทีละบรรทัด (command line) ซึ่งจะเป็นหน้าต่างที่เราใช้งานมากที่สุด
3. **Variable Browser** เป็นหน้าต่างที่แสดงรายละเอียดของตัวแปร (variable) แต่ละตัวที่อยู่ในหน่วยความจำ ณ ขณะนั้น

4. **Command History** เป็นหน้าต่างที่แสดงประวัติการเรียกใช้คำสั่งทั้งหมดใน Scilab Console ที่ผ่านมา โดยแสดงวันและเวลากำกับด้วย

โดยทั่วไปแล้ว เราสามารถพิมพ์คำสั่งต่าง ๆ ที่เราต้องการลงใน Scilab Console ได้โดยตรง เช่น ถ้าเราพิมพ์ “`1+2`” ลงใน Scilab Console เมื่อกด Enter จะได้ผลลัพธ์ดังนี้

```
-->1+2
ans  =
     3.
-->
```

นั่นคือ จะได้คำตอบเท่ากับ 3

อย่างไรก็ตาม ถ้าคราวนี้เราพิมพ์ว่า “`1+2;`” จะได้ผลลัพธ์ดังนี้

```
 -->1+2;
-->
```

ซึ่งดูเหมือนว่าจะไม่แสดงผลลัพธ์อันใดเลย แต่ที่จริงแล้ว Scilab จะยังทำการคำนวณหา 1+2 แล้วเก็บผลลัพธ์ที่ได้ (ซึ่งในที่นี้คือ 3) ไว้ในหน่วยความจำอยู่ (สังเกตจาก Variable Browser ที่แสดงว่ามีตัวแปรชื่อ ans อยู่) เพียงแต่ไม่แสดงผลลัพธ์ออกมาทาง Scilab Console เท่านั้น จึงเหมาะสำหรับเวลาที่เราต้องการซ่อนค่าของตัวแปรเอาไว้ เพื่อไม่ให้แสดงผลออกมาทาง Scilab Console มากจนเกินไป

สมมติว่าเราต้องการลบทุกตัวแปรออกจากหน่วยความจำ เราสามารถทำได้โดยการพิมพ์คำสั่ง `clear` ซึ่งจะมีผลให้ทุกตัวแปรที่เคยมีอยู่หายไป อย่างไรก็ตาม จะเห็นว่าคำสั่งนี้ไม่ได้ทำให้ Scilab Console ว่างลงแต่อย่างใด ซึ่งถ้าเราต้องการล้างเฉพาะหน้าต่าง Scilab Console ให้ว่างลง แต่ไม่ต้องการล้างค่าของตัวแปรใด ๆ เราสามารถทำได้โดยใช้คำสั่ง `clc`

ในขั้นต่อมา ให้เราพิมพ์คำสั่ง `clear` เพื่อล้างค่าของทุกตัวแปรในหน่วยความจำก่อน จะสังเกตเห็นว่า Variable Browser นั้นว่างลงด้วย จากนั้นให้พิมพ์คำสั่ง “`//1+2 This is a comment`” จะได้ผลลัพธ์ดังนี้

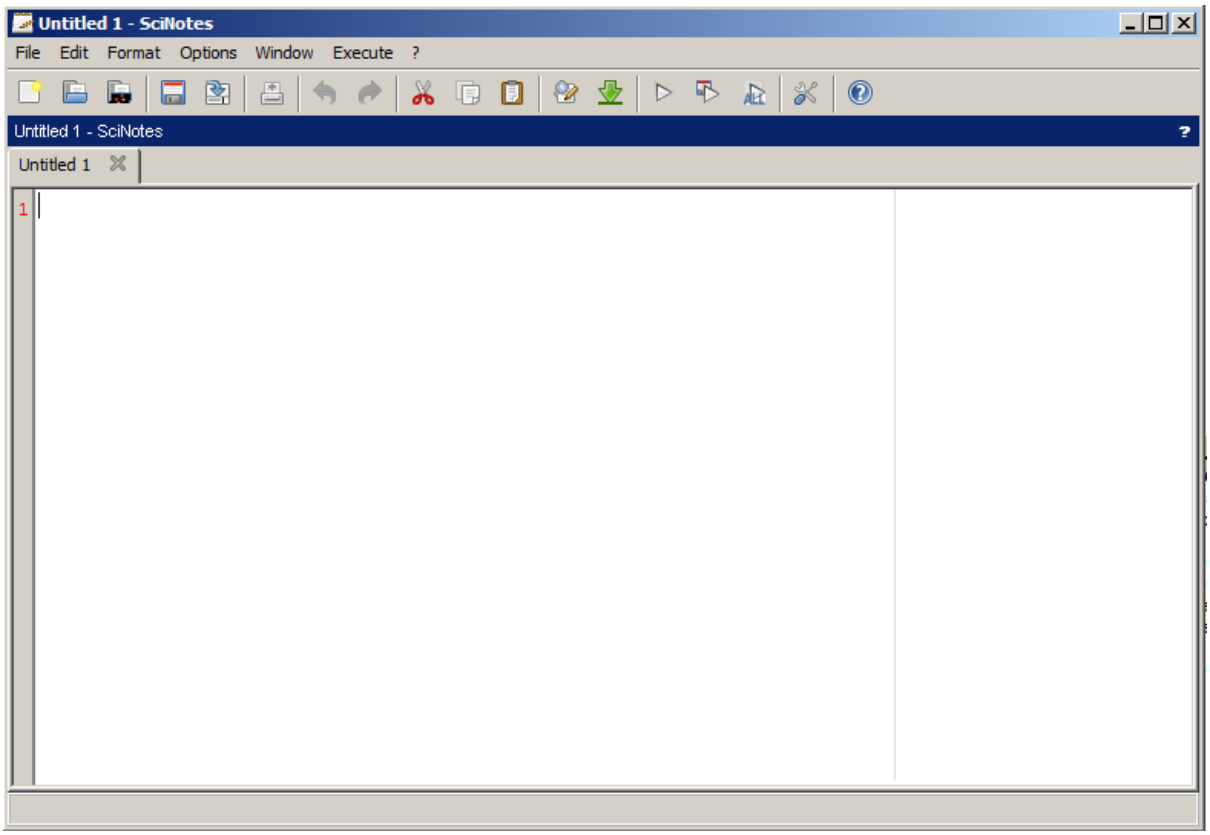
```
-->//1+2 This is a comment
-->
```

จะสังเกตเห็นว่าไม่มีการเปลี่ยนแปลงใน Variable Browser แต่อย่างใด นั่นคือ ไม่มีการคำนวณและเก็บค่าของตัวแปรใด ๆ ในหน่วยความจำเกิดขึ้น โดยทั่วไปแล้ว โปรแกรม Scilab จะไม่ทำตามคำสั่งใด ๆ ที่ขึ้นต้นด้วย “`//`” เราจึงสามารถเขียนคำอธิบายหรือหมายเหตุ (comment) ใด ๆ ในโปรแกรมของเราได้โดยใช้ “`//`” ขึ้นต้นบรรทัดที่เราต้องการเขียนเป็นคำอธิบาย

ถึงแม้ว่าเราสามารถพิมพ์คำสั่งใด ๆ ตามที่เราต้องการลงใน Scilab Console จะเห็นได้ว่าการแก้ไขคำสั่งที่เราได้พิมพ์ไปแล้วนั้นเป็นไปไม่ได้เลย นอกเสียจากว่าจะพิมพ์เป็นคำสั่งใหม่ที่ถูกต้องลงไป ซึ่งเป็นการเสียเวลาอย่างมาก นอกจากนี้ เมื่อเราออกจากโปรแกรม Scilab ไป ทุกคำสั่งที่เราเคยพิมพ์ไว้ใน Scilab Console ก็จะหายไปทั้งหมดด้วย ซึ่งเป็นอุปสรรคอย่างมากในการพัฒนาโปรแกรมที่มีความ

สลับซับซ้อนมากขึ้น ดังนั้น Scilab จึงมีทางออก ด้วยการให้ผู้ใช้งานสามารถพิมพ์คำสั่งทั้งหมดที่ต้องการไว้ในไฟล์ ๆ หนึ่งก่อน แล้วค่อย load ไฟล์ชุดคำสั่งนั้นเข้าสู่หน่วยความจำ แล้วค่อยทำการประมวลผลในคราวเดียว

ถ้าเราพิมพ์คำสั่ง `editor` ใน Scilab Console หรือคลิกที่ไอคอน Launce SciNotes บริเวณ Menu Bar ของโปรแกรม แล้วจะพบกับ SciNotes ซึ่งเป็น text editor ของโปรแกรม Scilab ดังรูป



รูปที่ 1.2 : หน้าต่างของ SciNotes

ใน SciNotes เราสามารถพิมพ์คำสั่งของโปรแกรม Scilab หลาย ๆ บรรทัดต่อกันได้ตามที่เราต้องการ แล้วค่อยบันทึก (save) เป็นไฟล์ไว้เพื่อเรียกใช้งานและแก้ไขได้ในภายหลัง เช่น ถ้าเราพิมพ์ชุดคำสั่งต่อไปนี้

```
disp(1+2)
1+2;
// 1+2 This is a comment
```

แล้วบันทึกเป็นไฟล์ชื่อ `test.sce` จากนั้นให้กดไอคอน execute บริเวณ Menu Bar ของ SciNotes จะได้ผลลัพธ์ปรากฏใน Scilab Console ดังนี้

```
-->exec('C:\Users\nookaussie\Documents\Teaching\323241\test.sce', -1)
     3.
```

สังเกตว่า 3 เป็นผลลัพธ์ที่เกิดมาจากคำสั่งในบรรทัดแรก

ในขั้นตอนสุดท้าย เมื่อเราต้องการออกจากโปรแกรม Scilab เราสามารถทำได้โดยการพิมพ์คำสั่ง `exit` หรือคลิกที่เครื่องหมายกากบาทตรงมุมบนขวาของหน้าต่างของโปรแกรม Scilab