

ใบงานการทดลองที่ 1

เรื่อง ภาษา Dart

1. จุดประสงค์

- 1 ใช้เครื่องมือสำหรับพัฒนาโปรแกรมภาษา Dart ได้
- 2 เขียนประกาศตัวแปร เงื่อนไขและลูปในภาษา Dart ได้

2. ทฤษฎี

บทที่ 1.2 ตัวแปร

1.2.1 ประเภทตัวแปรพื้นฐานในภาษา Dart

ในภาษา Dart มีประเภทตัวแปรพื้นฐานหลายประเภทที่ใช้ในการประกาศตัวแปรและเก็บข้อมูลต่าง ๆ สามารถดูได้ดังตารางที่ 1.2.1.1

ประเภท	ความหมาย
int	ใช้เก็บจำนวนเต็ม เช่น 1, -5, 1000
double	ใช้เก็บจำนวนทศนิยม เช่น 3.14, -0.5, 100.0
String	ใช้เก็บข้อความ เช่น "Hello, Dart", 'OpenAI', "12345"
bool	ใช้เก็บค่าความจริง (true หรือ false) เช่น 'true', 'false'
List	ใช้เก็บชุดข้อมูลที่เรียงต่อกัน เช่น [1, 2, 3], ["apple", "banana", "cherry"]
Map	ใช้เก็บข้อมูลแบบ key-value pairs เช่น {"name": "John", "age": 30, "city": "New York"}
dynamic	สามารถเก็บค่าข้อมูลประเภทใดก็ได้ (เปลี่ยนแปลงประเภทได้เรื่อย ๆ)
var	สามารถเก็บค่าข้อมูลประเภทใดก็ได้ (ไม่สามารถเปลี่ยนแปลงประเภทได้หลังจากรับประเภทตัวแปรแรก)

ตารางที่ 1.2.1.1 ตารางแสดงประเภทตัวแปรและความหมาย

1.2.2 รูปแบบการประกาศตัวแปร

ในการเขียนโปรแกรมคอมพิวเตอร์นั้น สิ่งที่นักเขียนโปรแกรมจะต้องรู้คือ รูปแบบการ ประกาศตัว เพื่อนำมาใช้ในการทำงาน ของโปรแกรมที่ได้เขียนขึ้นมา ในภาษาDartนั้น มีรูปแบบการประกาศตัวแปรดังต่อไปนี้

type variable_name;

หรืออีกรูปแบบหนึ่ง

type variable_name = value;

อธิบาย

type	คือ ชนิดของตัวแปรที่จะประกาศ ซึ่งควรจะมีการกำหนดชนิดของตัวแปรให้มีความเหมาะสมกับงานที่จะนำไปใช้
variable_name	คือ ชื่อของตัวแปรที่ต้องการสร้าง
value	คือ ค่าของตัวแปรนั้น ๆ ตามที่กำหนดให้

1.2.3 การประกาศและใช้งานตัวแปร

ตัวเลข (numbers)

ตัวเลขในภาษา Dart แบ่งออกเป็น 2 ประเภทหลักๆ คือ

จำนวนเต็ม (int)

- เป็นตัวเลขที่ไม่มีทศนิยม
- มีขอบเขตตั้งแต่ -2,147,483,648 ถึง 2,147,483,647

ตัวเลขทศนิยม (double)

- เป็นตัวเลขที่มีทศนิยม
- มีขอบเขตกว้างกว่าจำนวนเต็ม

```
int age = 25;
```

```
double height = 1.70;
```

```
num weight = 70.5;
```

ข้อความ (strings)

ข้อความในภาษา Dart เรียกว่า String สามารถสร้างข้อความได้โดยการใช้เครื่องหมายอัญประกาศคู่ หรือเครื่องหมายอัญประกาศสามคู่

ตัวอย่างการใช้ประเภทข้อมูลข้อความ

```
String name = "John Doe";
```

```
String message = "" Hello, world! "";
```

ตรรกะ (booleans)

ตรรกะในภาษา Dart เรียกว่า bool มีค่าได้เพียง 2 ค่า คือ true หรือ false

ตัวอย่างการใช้ประเภทข้อมูลตรรกะ

```
bool isStudent = false;
```

รายการ (lists)

รายการในภาษา Dart เรียกว่า List เป็นโครงสร้างข้อมูลแบบลำดับที่สามารถเก็บค่าข้อมูลได้หลายค่า ตัวอย่างการใช้ประเภทข้อมูลรายการ

```
List<int> numbers = [1, 2, 3, 4, 5];
```

```
List<String> names = ["John", "Doe", "Smith"];
```

ตัวแปรแบบไดนามิก (dynamic)

ตัวแปรแบบไดนามิกในภาษา Dart เรียกว่า dynamic สามารถเก็บค่าข้อมูลประเภทใดก็ได้ สามารถเปลี่ยนชนิดข้อมูลของตัวแปรแบบ dynamic ได้ตลอดเวลา

ตัวอย่างการใช้ประเภทข้อมูลตัวแปรแบบไดนามิก

```
dynamic value = 1;
```

```
dynamic value = "Hello, world!";
```

```
dynamic value = true;
```

ตัวแปรแบบ var

การประกาศตัวแปรแบบ var ในภาษา Dart เป็นการประกาศตัวแปรโดยไม่จำเป็นต้องกำหนดชนิดข้อมูลให้กับตัวแปร ตัวแปรแบบ var จะกำหนดชนิดข้อมูลให้กับตัวแปรโดยอัตโนมัติตามค่าที่ให้กับตัวแปรตอนประกาศ ไม่สามารถเปลี่ยนชนิดข้อมูลของตัวแปรแบบ var หลังจากที่กำหนดค่าให้กับตัวแปรแล้ว

ตัวอย่างการใช้ประเภทข้อมูลตัวแปรแบบ var

```
var name = "Teerasej";
```

```
var age = 25;
```

```
var isMale = true;
```

ข้อดีและข้อเสียในการประกาศตัวแปรแบบ var และ dynamic

การประกาศตัวแปรแบบ var

- ข้อดี: เขียนโค้ดได้สั้นและกระชับ
- ข้อเสีย: อาจทำให้โค้ดไม่ชัดเจนและเข้าใจยาก

การประกาศตัวแปรแบบ dynamic

- ข้อดี: ยืดหยุ่นและสามารถปรับเปลี่ยนโค้ดได้สะดวก
- ข้อเสีย: อาจทำให้โค้ดไม่ปลอดภัยและก่อให้เกิดช่องโหว่ด้านความปลอดภัย

การเลือกการประกาศตัวแปรแบบ var หรือ dynamic

ควรเลือกการประกาศตัวแปรแบบ var หรือ dynamic ให้เหมาะสมกับการใช้งาน โดยทั่วไปแล้ว ควรใช้การประกาศตัวแปรแบบ var หากมั่นใจว่าตัวแปรจะมีชนิดข้อมูลที่คงที่ตลอดอายุการใช้งาน และใช้การประกาศตัวแปรแบบ dynamic หากต้องการความยืดหยุ่นในการเก็บค่าของตัวแปร

บทที่ 1.3 เครื่องหมาย

เครื่องหมายในภาษา Dart มีความสำคัญมากในการเขียนโปรแกรม เนื่องจากใช้ในการดำเนินการทางคณิตศาสตร์ เปรียบเทียบ และตรรกะ ซึ่งจะช่วยให้เราสามารถเขียนโปรแกรมได้อย่างถูกต้องและมีประสิทธิภาพ เครื่องหมายในภาษา Dart มี 3 ประเภทหลักๆ ดังนี้

1.3.1 เครื่องหมายเชิงคณิตศาสตร์

เครื่องหมายเชิงคณิตศาสตร์ในภาษา Dart ใช้ในการดำเนินการทางคณิตศาสตร์ เช่น การบวก การลบ การคูณ การหาร และการหารส่วนเหลือ โดยสามารถดูสัญลักษณ์เครื่องหมายและความหมายได้ดังตารางที่ 1.3.1.1

เครื่องหมาย	ความหมาย
+	เครื่องหมายบวกใช้ในการบวกค่าตัวแปร
-	เครื่องหมายลบใช้ในการลบค่าตัวแปร
*	เครื่องหมายคูณใช้ในการคูณค่าตัวแปร
/	เครื่องหมายหารใช้ในการหารค่าตัวแปร
%	เครื่องหมายโมดูลัสใช้ในการหารค่าตัวแปรและคืนเศษ

ตารางที่ 1.3.1.1 ตารางแสดงเครื่องหมายเชิงคณิตศาสตร์และความหมาย

1.3.2 เครื่องหมายเชิงเปรียบเทียบ

เครื่องหมายเปรียบเทียบในภาษา Dart ใช้ในการเปรียบเทียบค่าสองค่า เช่น เท่ากับ ไม่เท่ากับ มากกว่า น้อยกว่า มากกว่าหรือเท่ากับ น้อยกว่าหรือเท่ากับ โดยสามารถดูสัญลักษณ์เครื่องหมายและความหมายได้ดังตารางที่ 1.3.2.1

เครื่องหมาย	ความหมาย
==	เครื่องหมายเท่ากับเท่ากันใช้ในการเปรียบเทียบว่าสองค่าเท่ากันหรือไม่
!=	เครื่องหมายไม่เท่ากับใช้ในการเปรียบเทียบว่าสองค่าไม่เท่ากัน
>	เครื่องหมายมากกว่าใช้ในการเปรียบเทียบค่าทางตัวเลขและตรรกะว่าค่าด้านซ้ายมากกว่าค่าด้านขวา
<	เครื่องหมายน้อยกว่าใช้ในการเปรียบเทียบค่าทางตัวเลขและตรรกะว่าค่าด้านซ้ายน้อยกว่าค่าด้านขวา
>=	เครื่องหมายมากกว่าหรือเท่ากับใช้ในการเปรียบเทียบค่าทางตัวเลขและตรรกะว่าค่าด้านซ้ายมากกว่าหรือเท่ากับค่าด้านขวา
<=	เครื่องหมายน้อยกว่าหรือเท่ากับใช้ในการเปรียบเทียบค่าทางตัวเลขและตรรกะว่าค่าด้านซ้ายน้อยกว่าหรือเท่ากับค่าด้านขวา

ตารางที่ 1.3.2.1 ตารางแสดงเครื่องหมายเปรียบเทียบและความหมาย

1.3.3 เครื่องหมายเชิงตรรกะ

เครื่องหมายตรรกะในภาษา Dart ใช้ในการดำเนินการทางตรรกะ เช่น AND OR NOT โดยสามารถดูสัญลักษณ์เครื่องหมายและความหมายได้ดังตารางที่ 1.3.3.1

เครื่องหมาย	ความหมาย
&&	เครื่องหมายและใช้ในการรวมเงื่อนไขตรรกะและควบคุมการสร้างเงื่อนไขที่ต้องเป็นจริงทั้งสองข้าง
	เครื่องหมายหรือใช้ในการรวมเงื่อนไขตรรกะและควบคุมการสร้างเงื่อนไขที่ต้องเป็นจริงอย่างน้อยหนึ่งข้าง
!	เครื่องหมายตรรกะ NOT ใช้ในการเปลี่ยนค่าตรรกะจากจริงเป็นเท็จหรือจากเท็จเป็นจริง

ตารางที่ 1.3.3.1 ตารางแสดงเครื่องหมายตรรกะและความหมาย

บทที่ 1.4 เงื่อนไขและลูป

1.4.1 คำสั่งเงื่อนไข

โครงสร้างเงื่อนไขในภาษา Dart ใช้ในการกำหนดว่าโปรแกรมจะดำเนินการต่ออย่างไรขึ้นอยู่กับเงื่อนไขที่กำหนดไว้ โครงสร้างเงื่อนไขในภาษา Dart มี 2 ประเภทหลักๆ ดังนี้

- If - else
- Switch – case

คำสั่ง If - else

คำสั่ง if-else เป็นคำสั่งที่มีเงื่อนไขเดียว ทำงานเฉพาะเมื่อเงื่อนไขที่กำหนดไว้เป็นจริง หากเงื่อนไขเป็นเท็จ คำสั่งในบล็อก else จะถูกเรียกทำงาน

โครงสร้าง if-else มีรูปแบบดังนี้

```
if (เงื่อนไข) {
    // ทำคำสั่งหากเงื่อนไขเป็นจริง
} else {
    // ทำคำสั่งหากเงื่อนไขเป็นเท็จ
}
```

ตัวอย่างการใช้งาน

```
void main() {
    var a = 10;
    var b = 20;

    if (a > b) {
        print("a มากกว่า b");
    } else {
        print("a ไม่มากกว่า b");
    }
}
```

ผลลัพธ์
a ไม่มากกว่า b

ในตัวอย่างนี้ เราจะกำหนดตัวแปร a เป็นค่า 10 และตัวแปร b เป็นค่า 20 จากนั้นเราจะใช้โครงสร้างเงื่อนไข if-else เพื่อตัดสินใจว่าจะทำคำสั่งใดต่อไป โดยเงื่อนไขคือ $a > b$ ถ้าเงื่อนไขเป็นจริงก็จะพิมพ์ข้อความว่า "a มากกว่า b" แต่ถ้าเงื่อนไขเป็นเท็จก็จะพิมพ์ข้อความว่า "a ไม่มากกว่า b"

คำสั่ง switch-case

คำสั่ง switch-case เป็นคำสั่งที่มีเงื่อนไขหลายเงื่อนไข ทำงานเฉพาะเมื่อเงื่อนไขที่กำหนดไว้ตรงกับเงื่อนไขใน case หากเงื่อนไขใน case ใดตรงกัน คำสั่งในบล็อก case นั้นจะถูกเรียกทำงาน

โครงสร้างเงื่อนไข switch มีรูปแบบดังนี้

```
switch (ตัวแปร) {
    case ค่าที่ 1:
        // คำสั่งที่จะทำถ้าค่าของตัวแปรตรงกับค่าที่ 1
        break;
    case ค่าที่ 2:
        // คำสั่งที่จะทำถ้าค่าของตัวแปรตรงกับค่าที่ 2
        break;
    ...
    default:
        // คำสั่งที่จะทำถ้าค่าของตัวแปรไม่ตรงกับค่าใดๆ ที่กำหนดไว้ใน
        case
        break;
}
```

ตัวอย่างการใช้งาน

```
void main() {
  var number = "1";
  switch (number) {
    case "1":
      print("เป็นเลข 1");
      break;
    case "2":
      print("เป็นเลข 2");
      break;
    default:
      print("เลขไม่ตรงกับเลขอะไรเลย");
      break;
  }
}
```

ผลลัพธ์
เป็นเลข 1

ในตัวอย่างนี้ เราจะกำหนดตัวแปร number เป็นค่า "1" จากนั้นเราจะใช้โครงสร้างเงื่อนไข switch เพื่อตัดสินใจว่าจะทำคำสั่งใดต่อไป โดยค่าของตัวแปร number จะตรงกับค่าที่กำหนดไว้ใน case นั้นๆ ถ้าค่าของตัวแปร number ตรงกับค่าที่กำหนดไว้ก็จะทำคำสั่งที่อยู่ใน case นั้น แต่ถ้าค่าของตัวแปร number ไม่ตรงกับค่าที่กำหนดไว้ ก็จะทำคำสั่งที่อยู่ใน default คำสั่ง break ใช้ในโครงสร้างเงื่อนไข switch เพื่อหยุดการวนซ้ำของโครงสร้างเงื่อนไข

1.4.2 คำสั่งทำซ้ำ

คำสั่งทำซ้ำเป็นคำสั่งควบคุมที่ทำงานซ้ำๆ คำสั่งในโครงสร้างวนซ้ำจะถูกเรียกทำงานซ้ำๆ จนกว่าจะถึงเงื่อนไขที่กำหนดไว้

โครงสร้างวนซ้ำในภาษา Dart มี 3 รูปแบบ คือ

- คำสั่ง while
- คำสั่ง for
- คำสั่ง do-while

คำสั่ง while

โครงสร้างคำสั่ง while ในภาษา Dart ใช้ในการวนซ้ำคำสั่งต่างๆ ไปเรื่อยๆ ตราบที่เงื่อนไขที่กำหนดไว้เป็นจริง โครงสร้างคำสั่ง while มีรูปแบบดังนี้

```
while (เงื่อนไข) {
  // คำสั่งที่จะทำซ้ำๆ
}
```

ตัวอย่างการใช้งาน

```
void main() {
  var i = 1;

  while (i <= 5 ) {
    print(i);
    i++;
  }
}
```

ผลลัพธ์

```
1
2
3
4
5
```

ในตัวอย่างนี้ เราจะวนซ้ำคำสั่ง print() ไปเรื่อยๆ トラバที่ค่าของตัวแปร i น้อยกว่าหรือเท่ากับ 5 เมื่อค่าของตัวแปร i มากกว่า 5 เงื่อนไขจะเป็นเท็จ และโครงสร้างคำสั่ง while จะหยุดการวนซ้ำ

คำสั่ง for

โครงสร้างคำสั่ง for ในภาษา Dart ใช้ในการวนซ้ำคำสั่งต่างๆ ไปเรื่อย ๆ ตามจำนวนรอบที่กำหนด โครงสร้างคำสั่ง for มีรูปแบบดังนี้

```
for (ตัวแปรเริ่มต้น; เงื่อนไข; คำสั่งเปลี่ยนแปลงตัวแปร) {
  // คำสั่งที่จะทำซ้ำๆ
}
```

ตัวอย่างการใช้งาน

```
void main() {
  for (var i = 1; i <= 5; i++) {
    print(i);
  }
}
```

ผลลัพธ์

```
1
2
3
4
5
```

ในตัวอย่างนี้ เราจะวนซ้ำคำสั่ง print() ไปเรื่อย ๆ จำนวน 5 รอบ โดยเริ่มต้นค่าของตัวแปร i เป็น 1 จากนั้นจะเพิ่มค่าของตัวแปร i ที่ละ 1 ในแต่ละรอบ

คำสั่ง do – while

โครงสร้างคำสั่ง do-while ในภาษา Dart ใช้ในการวนซ้ำคำสั่งต่างๆ ไปเรื่อย ๆ อย่างน้อย 1 รอบ ตรวจสอบเงื่อนไขที่กำหนดไว้เป็นจริง โครงสร้างคำสั่ง do-while มีรูปแบบดังนี้

```
do {
    // คำสั่งที่จะทำซ้ำๆ
} while (เงื่อนไข);
```

ตัวอย่างการใช้งาน

```
void main() {
    var i = 1;

    do {
        print(i);
        i++;
    } while (i <= 5);
}
```

ผลลัพธ์

```
1
2
3
4
5
```

ในตัวอย่างนี้ เราจะวนซ้ำคำสั่ง print() ไปเรื่อย ๆ อย่างน้อย 1 รอบ โดยเริ่มต้นค่าของตัวแปร i เป็น 1 จากนั้นจะเพิ่มค่าของตัวแปร i ทีละ 1 ในแต่ละรอบ

3. เครื่องมือและอุปกรณ์การทดลอง

- 3.1 คอมพิวเตอร์ 1 เครื่อง
- 3.2 ใบงานที่ 1 เรื่อง คำสั่งเลือกเงื่อนไข

4. ลำดับขั้นการทดลอง

- 4.1 นักเรียนศึกษาเนื้อหา เรื่อง คำสั่งเลือกเงื่อนไข
- 4.2 ให้นักเรียนตอบคำถาม ลงในใบงานที่ 1
- 4.3 ส่งงานครูหลังจากเสร็จเรียบร้อยแล้ว

คำสั่ง ให้นักศึกษาเขียนคำตอบตามที่โจทย์กำหนดให้ถูกต้อง (สามารถแนบรูปโค้ดและผลลัพธ์คำตอบของโปรแกรมได้)

1. เขียนโปรแกรมคำนวณพื้นที่สามเหลี่ยมมุมฉาก

```
Run | Debug
1 void main() {
2     double area;
3     double base = 50;
4     double high = 20;
5     area = 1/2*base*high;
6     print('Area = $area');
7 }
```

ผลลัพธ์ที่ได้

```
PS C:\Users\Master> dart
Area = 500.0
```

2. หากมีการประกาศตัวแปร `var n1=10` และ `dynamic n2=5` ตามในกล่องข้อความ แต่มีการเปลี่ยนแปลงค่าตัวแปรที่ประกาศในภายหลัง เมื่อมีการ print ค่า `n1` และ `n2` นักศึกษาคิดว่าจะสามารถ print ออกมาได้หรือไม่ เพราะเหตุใด และผลลัพธ์เป็นอย่างไร

ประกาศตัวแปร

```
var n1 = 10;
```

```
dynamic n2 = 5;
```

เปลี่ยนค่าใหม่

```
n1 = 11.5;
```

```
n2 = 15.4;
```

```
print('$n1');
```

```
print('$n2');
```

ไม่สามารถ print ได้ เพราะ var เป็นตัวแปรที่ไม่สามารถเปลี่ยนค่าได้ ดังนั้น `n1` จึงไม่สามารถเปลี่ยนค่าในภายหลังได้ ผลลัพธ์ที่ได้จะ Error

3. หากมีการประกาศตัวแปร **var n1** และ **dynamic n2** พร้อมทั้งกำหนดค่าให้ในบรรทัดต่อมาเป็น **n1=10** และ **n2=5** ตามในกล่องข้อความ แต่มีการเปลี่ยนแปลงค่าตัวแปรที่ประกาศในภายหลัง เมื่อมีการ print ค่า n1 และ n2 นักศึกษาคิดว่าจะสามารถ print ออกมาได้หรือไม่ เพราะเหตุใด

ประกาศตัวแปร

```
var n1;
```

```
dynamic n2;
```

```
n1 = 10;
```

$$n_2 = 5;$$

เปลี่ยนค่าใหม่

n1 = 11.5;

$$n_2 = 15.4;$$

```
print('$n1');
```

```
print('$n2');
```

สามารถ print ได้ เพราะ ไม่ได้กำหนดค่าที่ตัวแปร var n1 แต่มากำหนดภายหลังจะมองเป็นตัวแปรแบบ dynamic เมื่อมีการเปลี่ยนค่าอีกภายหลัง ก็จะสามารถ print ค่าออกมาได้ปกติ

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

4. เขียนโปรแกรมตรวจสอบค่าตัวเลขจำนวนเต็ม (number) โดยหาว่าเป็นจำนวนเต็มบวก (Positive Integer), จำนวนเต็มศูนย์ (Zero Integer) หรือ จำนวนเต็มลบ (Negative Integer) โดยใช้ if else

```
Run | Debug
1 void main() {
2     int number = 50;
3
4     if (number > 0) {
5         print("จำนวนเต็มบวก (Positive Integer)");
6     } else if (number < 0) {
7         print("จำนวนเต็มลบ (Negative Integer)");
8     } else {
9         print("จำนวนเต็มศูนย์ (Zero Integer)");
10    }
11 }
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Master> dart "c:\Users\Master\Desktop\DevApp\lab1\Lab1-2.dart"
จำนวนเต็มบวก (Positive Integer)
PS C:\Users\Master>
```

```
Run | Debug
1 void main() {
2     int number = -100;
3
4     if (number > 0) {
5         print("จำนวนเต็มบวก (Positive Integer)");
6     } else if (number < 0) {
7         print("จำนวนเต็มลบ (Negative Integer)");
8     } else {
9         print("จำนวนเต็มศูนย์ (Zero Integer)");
10    }
11 }
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Master> dart "c:\Users\Master\Desktop\DevApp\lab1\Lab1-2.dart"
จำนวนเต็มลบ (Negative Integer)
PS C:\Users\Master>
```

```
Run | Debug
1 void main() {
2     int number = 0;
3
4     if (number > 0) {
5         print("จำนวนเต็มบวก (Positive Integer)");
6     } else if (number < 0) {
7         print("จำนวนเต็มลบ (Negative Integer)");
8     } else {
9         print("จำนวนเต็มศูนย์ (Zero Integer)");
10    }
11 }
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Master> dart "c:\Users\Master\Desktop\DevApp\lab1\Lab1-2.dart"
จำนวนเต็มศูนย์ (Zero Integer)
PS C:\Users\Master>
```

5. เขียนโปรแกรม สูตรคูณแม่ M โดยใช้ for

```
Run | Debug
1 void main() {
2     int M = 8;
3
4     for(int i = 1; i<=12; i++){
5         print("$M * $i = ${M*i}");
6     }
7 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Master> dart "c:\Users\Master\Desktop\DevApp\lab1\Lab1-3.dart"
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
8 * 11 = 88
8 * 12 = 96
PS C:\Users\Master>
```

6. เขียนโปรแกรม สูตรคูณแม่ M โดยใช้ while

```
Run | Debug
9  void main() {
10     int M = 12;
11     int i = 1;
12     while(i <=12 ){
13         print("$M * $i = ${M*i}");
14         i++;
15     }
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Master> dart "c:\Users\Master\Desktop\DevApp\lab1\Lab1-3.dart"
12 * 1 = 12
12 * 2 = 24
12 * 3 = 36
12 * 4 = 48
12 * 5 = 60
12 * 6 = 72
12 * 7 = 84
12 * 8 = 96
12 * 9 = 108
12 * 10 = 120
12 * 11 = 132
12 * 12 = 144
```


5. สรุปผลการทดลอง

จากการทดลอง พบว่า ภาษา Dart มีการเขียนที่คล้ายกับภาษา C/C++ หรือ Java script ทำให้สามารถทำความเข้าใจได้ง่าย

การกระทำคำสั่งที่มีเงื่อนไข หรือ วนลูป ก็มีความคล้ายคลึงกัน ภาษา Dart จะมีการประกาศตัวแปรเพิ่มเข้า คือ การประกาศตัวแปร var และ dynamic เป็นการประกาศตัวแปรที่สามารถเก็บค่าชนิดใดก็ได้ทำให้สะดวกต่อการใช้งานอย่างมาก โดยทั้งสองอย่างจะมีข้อแตกต่างกันเล็กน้อยที่ var ไม่สามารถเปลี่ยนค่าได้ในภายหลัง แต่ dynamic สามารถเปลี่ยนได้

6. คำถามหลังการทดลอง

6.1 var และ dynamic ต่างกันอย่างไร

dynamic สามารถเก็บข้อมูลประเภทไหนก็ได้ สามารถเปลี่ยนค่าได้ในภายหลัง แต่ var สามารถเก็บข้อมูลประเภทไหนก็ได้ แต่ไม่สามารถเปลี่ยนค่าได้ในภายหลัง

6.2 คำสั่ง if-else กับ switch-case ต่างกันอย่างไร

if-else จะใช้ในกรณีที่เงื่อนไขเดียว แต่ switch-case จะใช้ในกรณีที่มีหลายเงื่อนไข

6.3 while มีโครงสร้างการทำงานอย่างไร และหากเงื่อนไขไม่เป็นจริง loop จะยังทำงานอยู่หรือไม่

while(เงื่อนไข){

คำสั่งต่างๆ

}

หากเงื่อนไขไม่เป็นจริง loop จะหยุดทำงานทันที