# การเขียนโปรแกรม ด้วยภาษา C#

## Variables

# Variables

- ในทุกภาษาโปรแกรม จะต้องอนุญาตให้ผู้ใช้สามารถนำข้อมูลไปเก็บและเรียกใช้จากหน่วยความจำได้
  - พื้นที่เก็บข้อมูลในหน่วยความจำจะถูกเรียกตามชื่อตัวแปร (Variables)
  - คอมไพเลอร์จะรู้ว่าเราต้องการเนื้อที่ในหน่วยความจำมากน้อยเพียงใดในการเก็บข้อมูล
    - พิจารณาจาก type
  - ในภาษา C# นั้น ตัวแปรจะมีชนิดที่แน่นอน (Strong type)
    - เมื่อประกาศแล้วจะเปลี่ยนในภายหลังไม่ได้

# ประเภทของ Variables ในภาษา C#

## Local variable

- เก็บข้อมูลชั่วคราว (ในขอบเขต method)
- ไม่เป็นสมาชิกของ type

## Field

- เก็บข้อมูลภายในของ type
- เป็นสมาชิกของ type

## Parameter

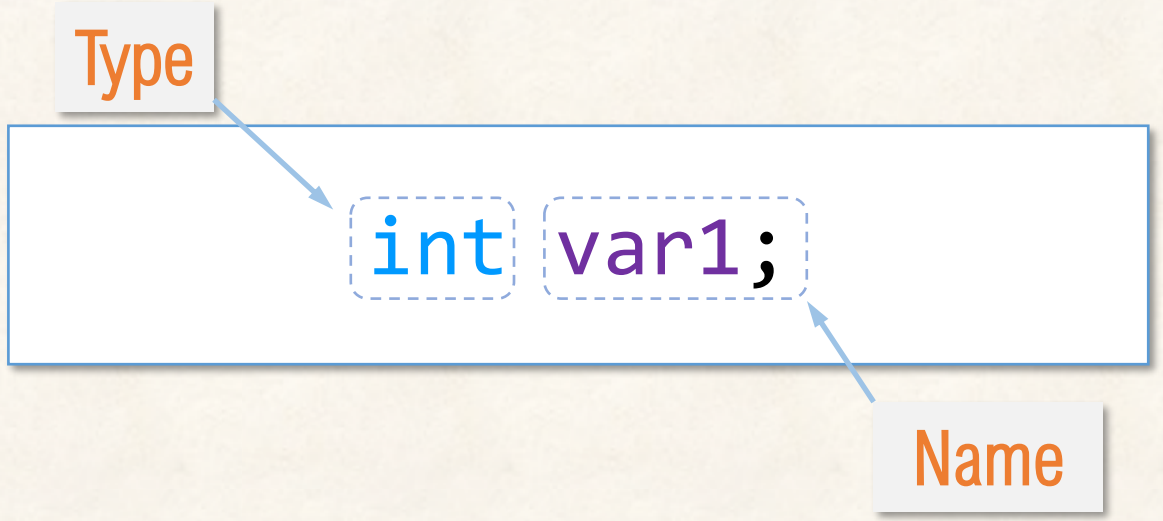
- เก็บข้อมูลชั่วคราว (เพื่อส่งให้ method)
- ไม่เป็นสมาชิกของ type

## Array element

- เก็บข้อมูลในลักษณะ collection
- เป็นสมาชิกของ type หรือไม่ก็ได้

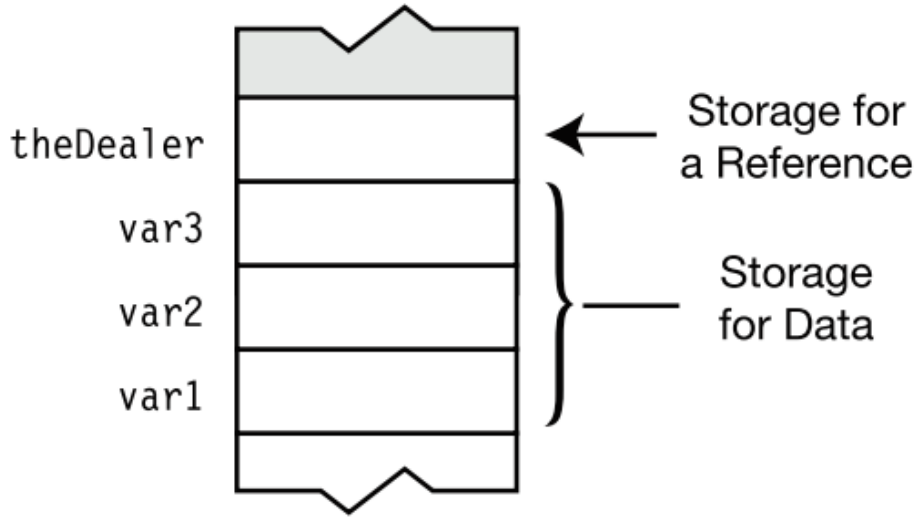
# Variables declaration

- ในภาษา C# นั้น ก่อนจะใช้งาน variables ได้ ต้องมีการ declare (ประกาศตัวแปร) ก่อน
  - เพื่อระบุชื่อ (Name) และชนิด (Type) ของ variable
  - เพื่อให้ compiler จัดเตรียมเนื้อที่ในหน่วยความจำได้ตามที่เราต้องการ



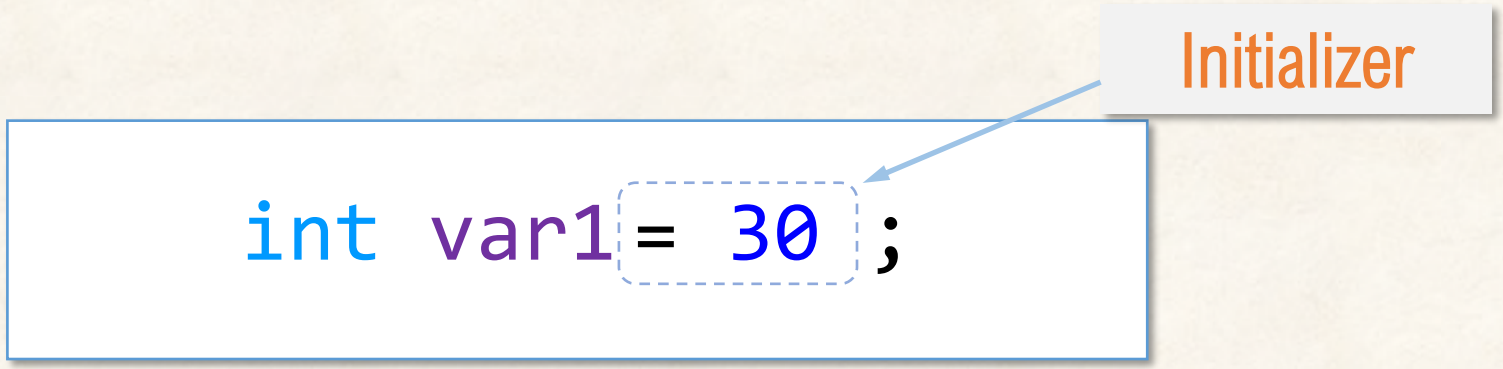
# ตัวอย่าง Variables declaration

```
int      var1;        // Value Type
int      var2;        // Value Type
float    var3;        // Value Type
Dealer   theDealer;   // Reference Type
```



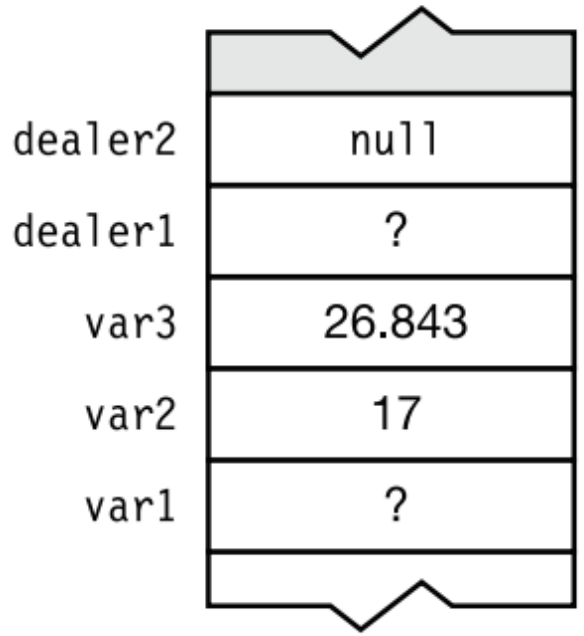
# Variable Initializers

- ในขณะที่ทำ Variable declaration เราสามารถกำหนดค่าเริ่มต้น (initialize) ให้ variable
  - ใช้เครื่องหมาย = ในการกำหนดค่าเริ่มต้น
  - Local variable ที่ไม่ได้กำหนดค่าเริ่มต้น จะมีสถานะเป็น undefined และไม่สามารถใช้งานได้ (compiler จะแจ้งข้อผิดพลาดขณะคอมไพล์)



# Variable Initializers

```
int     var1;              // Value Type
int     var2 = 17;         // Value Type
float   var3 = 26.843F;    // Value Type
Dealer  dealer1;           // Reference Type
Dealer  dealer2 = null;    // Reference Type
```



# Automatic Initialization

- Variable บางชนิดจะได้รับ initialization จากคอมไพเลอร์โดยอัตโนมัติ ในขณะที่บางชนิดเราต้องทำ initialization เอง

| Variable | Stored In | Auto-initialized | Use |
|---|---|---|---|
| Local variables | Stack or stack and heap | No | Used for local computation inside a function member |
| Class fields | Heap | Yes | Members of a `class` |
| Struct fields | Stack or heap | Yes | Members of a `struct` |
| Parameters Stack | | No | Used for passing values into and out of a method |
| Array elements | Heap | Yes | Members of an array |

# Multiple-Variable Declarations

- ในภาษา C#  เราสามารถประกาศตัวแปรหลายๆ ตัวไว้ในบรรทัดเดียวกันได้
  - ตัวแปร (Variables) เหล่านั้นต้องมีชนิด (Type) เดียวกันทั้งหมด
  - เราต้องแยกประกาศตัวแปรแต่ละตัวด้วยเครื่องหมายจุลภาค ( , )
  - ตัวแปรแต่ละตัว สามารถมี initializer ในขณะประกาศได้

# Multiple-Variable Declarations

OK, สามารถประกาศตัวแปรพร้อม initializer ในบรรทัดเดียวกันได้

```
int var1 = 30;
int var2, var3 = 30, var4, var5 = -4;
```

```
int var6, char var7, float var8;
```

Type

Difference type

Error, ไม่สามารถประกาศหลาย type ในบรรทัดเดียวได้

# Using the Value of a Variable

- Variable คือชื่อตัวแปรที่อ้างถึงค่าที่เก็บในหน่วยความจำ
  - เราสามารถนำค่านั้นมาใช้โดยอ้างผ่านชื่อตัวแปร

```
int var1 = 30;
Console.WriteLine($“var1 = {var1}”);
```

# Static Typing and the dynamic Keyword

- ในภาษา C# ทุกตัวแปรจะต้องมีการกำหนดชนิดก่อนนำไปใช้งาน
  - คอมไพเลอร์จะต้องรู้จักชนิดและจัดสรรตำแหน่งในหน่วยความจำให้เรา
  - คอมไพเลอร์จะต้องรู้ว่าควรเก็บตัวแปรนั้นบน stack หรือ heap
  - เราเรียกการใช้งานตัวแปรในลักษณะนี้ว่า static typing
- ในการเขียนโปรแกรม OOP ภาษาอื่น ๆ อาจจะมีการใช้งานตัวแปรแบบ dynamic typing เช่นภาษา IronPython หรือ IronRuby
- ในการเรียกใช้ Assemblies ที่เขียนด้วยภาษาอื่น ๆ เราอาจจะไม่สามารถระบุ type ของตัวแปรไว้ล่วงหน้าได้เสมอไป

# Static Typing and the dynamic Keyword

- ภาษา C# อนุญาตให้ใช้ dynamically typed
  - โดยการใช้คีย์เวิร์ด `dynamic`
  - คอมไพเลอร์จะไม่ตรวจสอบชนิดในตอนคอมไพล์ แต่จะรวบรวม information ต่าง ๆ เกี่ยวกับตัวแปรนั้นไว้ใน package ที่จะใช้ run โปรแกรม
  - ในตอน runtime จะมีการตรวจสอบชนิดของตัวแปรว่าเข้ากันได้กับตัวแปรใน assembly หรือไม่ ถ้าไม่ได้จะมีการ throw exception

# Nullable Types

- C# เป็นภาษาโปรแกรมที่ยืดหยุ่น สามารถใช้งานได้หลากหลาย
  - ตัวอย่างเช่นงานด้านฐานข้อมูล ถึงจะรู้จักชนิดของข้อมูลในแต่ละ field แต่ในบางครั้งข้อมูลที่อ่านได้อาจจะเป็นค่าที่ไม่ถูกต้อง (invalid) หรือยังไม่มีอยู่จริง (null)
  - เพื่อให้รองรับสถานการณ์ดังกล่าว type ใน C# จะมีการเพิ่ม Boolean indicator ที่เป็นตัวบอกว่าข้อมูลในตัวแปรนั้นอยู่ในสถานะ valid หรือไม่
  - Nullable type เป็นการสร้าง type ที่มีความสามารถระบุ (marked) ได้ว่าค่าในตัวแปรนั้น valid หรือ invalid