บทที่ 3
เรื่อง ตัวดำเนินการและนิพจน์ (Operator and Expressions)

วัตถุประสงค์

1. เพื่อให้นักศึกษาเข้าใจหลักการทำงานของตัวดำเนินการ (Operator) ประเภทต่างๆของภาษา C++
2. เพื่อให้นักศึกษาเข้าใจความหมายและสามารถสร้างนิพจน์ (Expressions) จากตัวดำเนินการต่างๆ ได้
3. เพื่อให้นักศึกษาสามารถใช้งานนิพจน์ (Expression) ร่วมกับคำสั่งอื่นได้ตามต้องการ

นิพจน์ (Expressions)

เป็นการสร้างนิพจน์ที่แทนข้อมูล 1 ค่า อาจเป็นตัวเลข ตัวอักษร โดยการนำข้อมูลมากระต่อกันตั้งแต่หนึ่งจำนวนขึ้นไป ซึ่งอาจเป็นข้อมูลเดียว หรือ หลายๆข้อมูลก็ได้ เช่น ตัวแปร ค่าคงที่ หรือฟังก์ชัน เป็นต้น แล้วนำข้อมูลมากระทำกันด้วยตัวดำเนินการ ซึ่งผลลัพธ์ที่ได้จะขึ้นอยู่กับตัวดำเนินการ

โดยข้อมูลที่นำมากระทำด้วยตัวดำเนินการ จะถูกเรียกว่า ตัวถูกดำเนินการ(Operand) เช่น

a + b x == y c + 3- a + b x / 4 – y (5 – a) % 4 ++i

ตัวดำเนินการ(Operators)

สำหรับตัวดำเนินการในภาษา C++ มีหลายแบบ สามารถแบ่งออกเป็นกลุ่มได้ดังนี้

1. ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

เป็นแบบ Binary Operator คือกระทำกับตัวถูกดำเนินการ(Operand) จำนวน 2 ตัวเสมอ โดยภาษา C++ มีตัวดำเนินการทางคณิตศาสตร์ 5 ตัว ดังนี้

ตัวดำเนินการ	วัตถุประสงค์
+	การบวก(addition)
-	การลบ(subtraction)
*	การคูณ(multiplication)
/	การหาร(division)
%	การหารเอาเศษจำนวนเต็ม(modulus)

ตัวถูกดำเนินการ(Operand) เป็นได้ทั้งค่าคงที่ ตัวแปร ฟังก์ชั่น หรือนิพจน์

หมายเหตุ เฉพาะตัวดำเนินการ % ชนิดข้อมูลแบบตัวเลขจำนวนเต็มเท่านั้น เช่น short, int, long ถ้าเอามาใช้กับตัวเลขจำนวนทศนิยม ก็จะเกิด Syntax error

รูปแบบการเขียนนิพจน์ประกอบด้วย operand 2 ตัว และ operator 1 ตัว

(Operand Operator Operand)

เช่น a + 5 H – W P / 4 A % 2 4 * 3

เมื่อใช้ตัวดำเนินการคณิตศาสตร์ทำงานระหว่าง operand ที่มีชนิดข้อมูลต่างกัน ผลลัพธ์ที่ได้จะเป็นชนิดข้อมูลที่มีขนาดใหญ่สุด เช่น ชนิดข้อมูล int กับ long ผลลัพธ์ก็จะเป็น long ซึ่งสามารถสรุปได้ตามตาราง

นิพจน์แรก	นิพจน์สอง	ผลลัพธ์จากการใช้ตัวดำเนินการคณิตศาสตร์
short	short	short
int	short	int
long	short หรือ int	long
short หรือ int	float	float
int	double	double
long	long double	long double

หมายเหตุ ชนิดข้อมูลในนิพจน์แรกและนิพจน์สองสามารถสลับกันได้

ตารางแสดงการทำงานของ Operator

หน้าที่	ตัวดำเนินการ	ตัวอย่าง	ผลลัพธ์
การบวก	+	2 + 3; 5 + 10;	5, int 15, int
การลบ	-	13 – 4; 4 – 7;	9, int -3, int
การคูณ	*	3 * 4; 5 * 11;	12, int 55, int
การหาร	/	8 / 2; 6 / 4; 11 / 4; 4 / 5; 6 / 0;	4 , int 1, int 2 , int 0, int Error divide by zero
การหารเอาเศษ	%	10 % 3; 23 % 4; 5 % 0;	1 , int 3 , int Error divide by zero

เมื่อมีการใช้ตัวดำเนินการหลายๆตัวในหนึ่งนิพจน์ได้ เพื่อให้การทำงานนั้นสามารถทำได้ถูกต้อง จึงต้องมีการกำหนดลำดับความสำคัญและรูปแบบการทำงานไว้ โดยตัวดำเนินการทางคณิตศาสตร์มีลำดับความสำคัญและการทำงาน ดังนี้

ตารางแสดงลำดับความสำคัญของตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการ	หน้าที่	การทำงาน
* / %	การคูณ การหาร การหารเอาเศษ	ซ้ายไปขวา
+ -	การบวก การลบ	ซ้ายไปขวา

เช่น 2 / 3 + 5, 8 - 7 % 2, 3 * 5 / 2

การทดลองที่ 3_1 โปรแกรมแสดงการทำงานของตัวดำเนินการทางคณิตศาสตร์

```
1  /*  Program 3_1 : Use arithmetic operators   */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      cout << "Result of 3 / 2 + 5 = " << 3 / 2 + 5 << endl;
7      cout << "Result of 5 / 1 + 2 = " << 5 / 1 + 2 << endl;
8      cout << "Result of 3 * 2 + 4 * 5 = " << 3 * 2 + 4 * 5 << endl;
9      cout << "Result of 4 - 2 + 5 / 3 + 2 = " << 4 - 2 + 5 / 3 + 2 << endl;
10     cout << "Result of 10 - 2 + 7 % 2 - 1 = " << 10 - 2 + 7 % 2 - 1 << endl;
11     return (0);
12 }
```

บันทึกผลการทดลอง

ให้นักศึกษาแก้ไขโปรแกรมโดยการใส่เพิ่มวงเล็บไว้ที่ตัวดำเนินการตัวแรก และดูผลลัพธ์

คำถาม ฉะนั้นการใส่วงเล็บในนิพจน์จะมีผลต่อการคำนวณของนิพจน์นั้นให้เปลี่ยนไปทุกครั้งหรือไม่

การทดลองที่ 3_2 โปรแกรมแสดงคำนวณหาค่าของสมการ $f(x) = x^3 + 3x - 10$

```
1  /*  Program 3_2 : compute eqation = X*X*X + 3*X - 10   */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          float X;
7          cout << "Enter value X : ";
8          cin >> X ;
9          float Result (0);
10         Result = X * X * X + 3 * X - 10;
11         cout << "\nResult of X*X*X + 3*X - 10 = ";
12         cout << Result << endl;
13         return (0);
14 }
```

บันทึกผลการทดลอง

2. ตัวดำเนินการเชิงสัมพันธ์และเชิงตรรกะ (Relational and Logical Operators)

เป็นแบบ Binary และ Unary Operator โดยตัวดำเนินการประเภทนี้จะให้ผลลัพธ์การดำเนินการมาเป็นค่าทางตรรกศาสตร์ คือเป็นค่าจริง(true)หรือค่าเท็จ(false)

โดยในภาษา C++ ค่าจริง(true) คือ ทุกค่าที่ไม่ใช่ 0 ให้ผลลัพธ์เป็น 1
ค่าเท็จ(false) คือ 0 ให้ผลลัพธ์เป็น 0

ตัวดำเนินการเชิงสัมพันธ์	หน้าที่
<	น้อยกว่า
<=	น้อยกว่า หรือเท่ากับ
>	มากกว่า
>=	มากกว่าหรือเท่ากับ
==	เท่ากับ
!=	ไม่เท่ากับ

ตัวดำเนินการเชิงตรรกศาสตร์	หน้าที่
!	ไม่
&&	และ
||	หรือ

สำหรับการทำงานของตัวดำเนินการนี้จะให้ผลลัพธ์เป็นค่าจริงหรือค่าเท็จ ซึ่งจะเท่ากับชนิดข้อมูล bool โดยมีการนำไปใช้ร่วมกับคำสั่งเงื่อนไขและคำสั่งการวนรอบ

ตารางแสดงการทำงานของตัวดำเนินการเชิงสัมพันธ์และเทียบเท่า

ตัวดำเนินการ	หน้าที่	ตัวอย่าง a = 5; b = 3;	ผลลัพธ์
==	เปรียบเทียบว่าต้องเท่ากัน	a == b	false
!=	เปรียบเทียบว่าต้องไม่เท่ากัน	a != b	true
<	เปรียบเทียบว่าต้องน้อยกว่า	a < b	false
<=	เปรียบเทียบว่าต้องน้อยกว่าหรือเท่ากับ	a <= b	false
>	เปรียบเทียบว่าต้องมากกว่า	a > b	true
>=	เปรียบเทียบว่าต้องมากกว่าหรือเท่ากับ	a >= b	true

ตารางแสดงการทำงานของตัวดำเนินการเชิงตรรกศาสตร์

A	B	! A	! B	A && B	A || B
false	false	true	true	false	false
false	true	true	false	false	true
true	false	false	true	false	true
true	true	false	false	true	true

การทดลองที่ 3_3 โปรแกรมแสดงการทำงานของตัวดำเนินการเชิงสัมพันธ์ และ เชิงตรรกศาสตร์

```
1  /*   Program 3_3 : Use relational and logical operator   */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          int       A, B;
7          cout << "Enter first number(A) : ";
8          cin >> A;
9          cout << "Enter second number(B) : ";
10         cin >> B;
11         cout << "\nTesting usage operator\n";
12         cout << "======================\n";
13         cout << " A <  B  = " << (A < B) << endl;
14         cout << " A <= B  = " << (A <= B) << endl;
15         cout << " A >  B  = " << (A > B) << endl;
16         cout << " A >= B  = " << (A >= B) << endl;
17         cout << " A == B  = " << (A == B) << endl;
18         cout << " A != B  = " << (A != B) << endl;
19         cout << " A && B  = " << (A && B) << endl;
20         cout << " A || B  = " << (A || B) << endl;
21         cout << "!A = " << !A << ",  !B  = " << !B << endl;
22         return (0);
23 }
```

บันทึกผลการทดลอง

3. ตัวดำเนินการยูนารี (Unary Operators)

เป็นแบบใช้ Operand เพียงตัวเดียว มีตัวดำเนินการยูนารี ดังต่อไปนี้

ตัวดำเนินการ	หน้าที่
-	ให้ค่าลบ(unary minus)
--	การลดค่าที่ละหนึ่ง(decrement)
++	การเพิ่มค่าที่ละหนึ่ง(increment)

รูปแบบการเขียนนิพจน์ ประกอบด้วย operand 1 ตัว และ operator 1 ตัว

(Operator Operand) Prefix หรือ (Operand Operator) Postfix

โดยปกติแล้วการเขียนตัวดำเนินการไว้ข้างหน้าหรือข้างหลังจะมีความหมายเท่ากับการเพิ่มหรือลดค่าไปหนึ่งของตัวแปรนั้นๆ แต้ถ้าหากมีการนำไปใช้ร่วมกับนิพจน์อื่น การทำงานก็จะเปลี่ยนไป เช่น

-a	หมายถึงการทำให้ค่าของตัวแปร a เป็นลบ
++a	หมายถึงการเพิ่มค่าให้กับตัวแปร a ขึ้นหนึ่งค่า
a++	หมายถึงการเพิ่มค่าให้กับตัวแปร a ขึ้นหนึ่งค่า
--a	หมายถึงการลดค่าให้กับตัวแปร a ลงหนึ่งค่า
a--	หมายถึงการลดค่าให้กับตัวแปร a ลงหนึ่งค่า

++a - 2	หมายถึงการคำนวณโดยนำค่าตัวแปร a เพิ่มหนึ่งค่าแล้วนำไปลบกับค่า 2
a++ - 2	หมายถึงการคำนวณโดยนำค่าตัวแปร a แล้วนำไปลบกับค่า 2
--a + 2	หมายถึงการคำนวณโดยนำค่าตัวแปร a ลดหนึ่งค่าแล้วนำไปบวกกับค่า 2
a-- + 2	หมายถึงการคำนวณโดยนำค่าตัวแปร a แล้วนำไปบวกกับค่า 2

ตัวอย่าง

ก่อนใช้	i				j			
นิพจน์	++i + ++j		++i - j++		--i + j--		i-- - --j	
ค่า								
หลังใช้	i	j	i	j	i	j	i	j

การทดลองที่ 3_4 โปรแกรมแสดงการทำงานของตัวดำเนินการเพิ่มค่า และลดค่า

```
 1  /*  Program 3_4 : Use increment and decrement operator   */
 2  #include <iostream>
 3  using namespace std;
 4  int main()
 5  {
 6          int       A, B;
 7          cout << "Enter first number(A) : ";
 8          cin >> A;
 9          cout << "Enter second number(B) : ";
10          cin >> B;
11          cout << endl;
12          cout << " Before A = " << A << endl;
13          cout << " ++A = " << ++A << endl;
14          cout << " After A = " << A << endl;
15          cout << " Before A = " << A << endl;
16          cout << " A++ = " << A++ << endl;
17          cout << " After A = " << A << endl << endl;
18          cout << " Before B = " << B << endl;
19          cout << " --B = " << --B << endl;
20          cout << " After B = " << B << endl;
21          cout << " Before B = " << B << endl;
22          cout << " B-- = " << B-- << endl;
23          cout << " After B = " << B << endl;
24          return(0);
25  }
```

บันทึกผลการทดลอง

4. ตัวดำเนินการกำหนดค่า (Assigment Operators)

ตัวดำเนินการในการกำหนดค่า lvalue หรือกำหนดค่าเริ่มต้นให้แก่ตัวแปร มีรูปแบบ ดังนี้

ตัวดำเนินการ	รูปแบบ
=	identifier = expression identifier 1 = identifier 2 = ... = expression
+=	expression 1 += expression 2
-=	expression 1 -= expression 2
*=	expression 1 *= expression 2
/=	expression 1 /= expression 2
%=	expression 1 %= expression 2

ตารางแสดงการทำงานของตัวดำเนินการกำหนดค่า

ตัวดำเนินการ	ความหมายและการใช้	ผลลัพธ์
=	x = 10; หมายถึงกำหนดค่าให้ lvalue	x มีค่า 10
+=	x += 1; หมายถึง x = x + 1;	x มีค่า 11
-=	x -= 1; หมายถึง x = x - 1;	x มีค่า 10
*=	x *= 3; หมายถึง x = x * 3;	x มีค่า 30
/=	x /= 3; หมายถึง x = x / 3;	x มีค่า 10
%=	x %= 3; หมายถึง x = x % 3;	x มีค่า 1

การทดลองที่ 3_5 โปรแกรมแสดงการทำงานของตัวดำเนินการกำหนดค่า

```
1  /*    Program 3_5 : Use assignment operator    */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          int  X = 10;
7          cout << "X = " << X << endl;
```

```
 8          X += 1;
 9          cout << "X += 1; \tX = 10 + 1 = " << X << endl;
10          X -= 1;
11          cout << "X -= 1; \tX = 11 + 1 = " << X << endl;
12          X *= 3;
13          cout << "X *= 3; \tX = 10 * 3 = " << X << endl;
14          X /= 3;
15          cout << "X /= 3; \tX = 30 / 3 = " << X << endl;
16          X %= 3;
17          cout << "X %= 3; \tX = 10 % 3 = " << X << endl;
18          return (0);
19  }
```

บันทึกผลการทดลอง

5. ตัวดำเนินการเงื่อนไข (Conditional operators)

ตัวดำเนินการนี้บางที เรียกว่า ตัวดำเนินการเทอรนารี่(Ternary Operator) เพราะว่าใช้นิพจน์ถึงสามนิพจน์ในการทำงาน ซึ่งตัวดำเนินการจะทำงานเหมือนคำสั่ง if-else อย่างง่ายๆได้ โดยมีรูปแบบของนิพจน์เงื่อนไขแสดงดังต่อไปนี้

expression1 ? expression 2 : expression 3

มีรูปแบบการทำงานคือ

ถ้า expression1 มีค่าเป็นจริง (คือไม่เป็น 0) จะทำงานตาม expression 2

ถ้า expression1 มีค่าไม่เป็นจริง (คือมีค่า 0) จะทำงานตาม expression 3

ตัวอย่าง

```
flag  =  (i < 0) ? 0 : 100
flag  =  (f >= g) ? f : g
```

การทดลองที่ 3_6 โปรแกรมแสดงการทำงานของตัวดำเนินการเงื่อนไข

```
1  /*   Program 3_6 : Use condition operator   */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          int  Number;
7          cout << "Enter number : " ;
8          cin >> Number;
9          cout << Number << " is " << ((Number % 2) == 0 ? "even " : "odd ") ;
10         cout << " number." << endl;
11         return (0);
12 }
```

บันทึกผลการทดลอง

6. ตัวดำเนินการระดับบิต (Bitwise operators)

ตัวดำเนินการนี้จะทำงานกับข้อมูลในลักษณะแบบบิตเป็นเลขฐานสอง เป็นตัวดำเนินการแบบไบนารี่ (Binary Operator) สามารถทำงานได้เร็วกว่า โดยมีตัวดำเนินการดังนี้

ตัวดำเนินการ	ความหมาย	รูปแบบการใช้
>>	Bit shift right เป็นการเลื่อนบิตไปทางขวา โดยการเติม 0 ไปทางซ้าย	x >> 1
<<	Bit shiftleft เป็นการเลื่อนบิตไปทางซ้าย โดยการเติม 0 ไปทางขวา	x << 1
&	Bitwise AND เป็นการ AND บิต	x & 4
|	Bitwise OR เป็นการ OR บิต	x | 3
^	Bitwise XOR เป็นการ XOR บิต	x ^ 5

ตัวอย่าง

```
int  x = 4;
x = x << 1;      // x = x * 2;
x = x >> 1;      // x = x / 2;
```

การทดลองที่ 3_7 โปรแกรมแสดงการทำงานของตัวดำเนินการระดับบิต

```
1  /*   Program 3_7 : Use Bitwise operator   */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          int n;
7          cout << "Enter integer number : " ;
8          cin >> n;
9          cout << endl;
10         cout << "n = " << n << endl;
11         cout << "n >> 1 = " << (n >> 1) << endl;
12         cout << "n << 1 = " << (n << 1) << endl;
13         cout << "n & 8 = " << (n & 8) << endl;
14         cout << "n | 15 = " << (n | 15) << endl;
15         cout << "n ^ 10 = " << (n ^ 10) << endl;
16         return(0);
17 }
```

บันทึกผลการทดลอง

การแปลงชนิดข้อมูล(Type Conversion)

การแปลงข้อมูล หมายถึง นิพจน์ชนิดข้อมูลหนึ่งถูกแปลงเป็นนิพจน์ข้อมูลอีกชนิดหนึ่ง ในการแปลงข้อมูลสามารถแบ่งออกเป็น 2 ชนิดใหญ่ๆ คือ การแปลงข้อมูลแบบมีนัย(Implicit type conversion) และการแปลงข้อมูลแบบชัดเจน(Explicit type conversion)

การแปลงข้อมูลแบบมีนัย(Implicit type conversion)

การแปลงข้อมูลแบบนี้จะเป็นการแปลงข้อมูลโดยพื้นฐานของตัวคอมไพล์ จะพบได้หลายกรณีดังนี้

1. นิพจน์คณิตศาสตร์ที่มีการใช้ชนิดข้อมูลต่างกันผสมกันอยู่ จะแปลงข้อมูลเป็นชนิดใหญ่ก่อนดำเนินการด้วยตัวดำเนินการ
   เช่น cout << 40 + 30.5 ;
2. ตัวดำเนินการกำหนดค่า จะแปลงข้อมูลให้เป็นชนิดเดียวกับที่อยู่ด้านซ้ายของตัวดำเนินการกำหนดค่า
   เช่น int x = 2.543;
3. การผ่านค่าอาร์กิวเมนต์ไปยังฟังก์ชั่น ซึ่งเป็นชนิดข้อมูลหนึ่งขณะที่พารามิเตอร์ของฟังก์ชั่นเป็นข้อมูลอีกชนิดหนึ่ง
   เช่น cout << sqrt(3) << endl;

การแปลงข้อมูลแบบชัดเจน(Explicit type conversion)

การแปลงชนิดข้อมูลนี้ต้องใช้ตัวดำเนินการชนิด cast ซึ่งในภาษ C++ จะมีรูปแบบการแปลงข้อมูลดังนี้

1. รูปแบบการใช้ตัวดำเนินการ cast แบบ ( ) exp
   (new-type) exp; หรือ new-type (exp) ;
   ตัวอย่าง x = (int) 4.5;

2. รูปแบบการใช้ตัวดำเนินการ cast แบบ static_cast
   static_cast<new-type> (exp);
   ตัวอย่าง x = static_cast<int> (4.5);

การทดลองที่ 3_8 โปรแกรมแสดงการทำงานของการแปลงชนิดข้อมูล

```
1  /*  Program 3_8 : Use type conversion   */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          int  Number1, Number2, Number3;
7          cout << "Enter first number : ";
8          cin >> Number1;
9          cout << "Enter second number : ";
10         cin >> Number2;
11         cout << "Enter third number : ";
12         cin >> Number3;
13         float Average;
14         Average = static_cast<float>(Number1 + Number2 + Number3) / 3;
15         cout << "\nAverage of 3 number = " << Average << endl;
16         return (0);
17 }
```

บันทึกผลการทดลอง

ตารางแสดงลำดับความสำคัญของตัวดำเนินการ

กลุ่มของตัวดำเนินการ	ตัวดำเนินการ	ลำดับการทำงาน
ตัวดำเนินการยูนารี	- ++ -- ! sizeof (type)	ขวาไปซ้าย
คูณ หาร และหาเศษเหลือ	* / %	ซ้ายไปขวา
บวกและลบ	+ -	ซ้ายไปขวา
Shift Bit Left, Shift Bit Right	<< >>	ซ้ายไปขวา
ตัวดำเนินการเชิงสัมพันธ์	< <= > >=	ซ้ายไปขวา
ตัวดำเนินการเทียบเท่า	== !=	ซ้ายไปขวา
ตัวดำเนินการ Bitwise AND	&	ซ้ายไปขวา
ตัวดำเนินการ Bitwise XOR	^	ซ้ายไปขวา
ตัวดำเนินการ Bitwise OR	|	ซ้ายไปขวา
Logical AND	&&	ซ้ายไปขวา
Logical OR	||	ซ้ายไปขวา
ตัวดำเนินการเงื่อนไข	? :	ขวาไปซ้าย
ตัวดำเนินการกำหนดค่า	= += -= *= /= %=	ขวาไปซ้าย

การทดลองที่ 3_9 โปรแกรมแสดงคำนวณหาภาษี

```
1  /*   Program 3_9 : Compute tax   */
2  #include <iostream>
3  #include <string>
4  using namespace std;
5  int main()
6  {
7          float    Price, Tax, Total, Tax_Rate = 0.07f;
8          string   ProductName;
9          cout << "Enter product name : ";
10         cin >> ProductName;
11         cout << "Enter product price : ";
```

```
12          cin >> Price;
13          cout << endl;
14          // case 1
15          Tax = Price * Tax_Rate;
16          Total = Price + Tax;
17          //case 2
18          //Total = Price + Price * Tax_Rate;
19          cout << "Price of " << ProductName << " = " << Price << endl;
20          cout << "Tax(%7) of " << ProductName << " = " << Tax << endl;
21          cout << "Total Price of " << ProductName << " = " << Total << endl;
22          return (0);
23  }
```

บันทึกผลการทดลอง

แบบฝึกหัดท้ายบท

1. ให้นักศึกษาหาผลลัพธ์ของคำสั่งภาษา C++ ต่อไปนี้ และบอกลำดับการทำงาน

1.1 int X = 7 + 3 * 6 / 2 - 1;

1.2 float Y = 2 % 2 + 2 * 2 - 2 / 2;

2. ให้นักศึกษาหาผลลัพธ์จากส่วนของโปรแกรมต่อไต่อไปนี้

```
int  m = 2;
int  n = 3
m = n + n;
n = m * 1.5;
cout << "m = " << m << "  n = " << n << endl;
```

3. ให้นักศึกษาเขียนโปรแกรมแปลงค่าอุณหภูมิจาก ฟาเร็นไฮน์เป็นเซลเซียส โดยให้รับค่าอุณหภูมิจากคีย์บอร์ด เป็น ฟาเร็นไฮน์ โดยมีสูตรในการคำนวณดังนี้ $C = \frac{5}{9}(F - 32)$

4. ให้นักศึกษาเขียนโปรแกรมรับค่าตัวเลขจำนวนเต็ม 1 ค่า จำนวน 4 หลักเสมอ แล้วทำการแสดงผลตัวเลขนั้น โดยแยกออกมาที่ละหลักโดยมีช่องว่างคั่นที่ละ 3 ตัว

```
Enter number : 4263
4   2   6   3
```

5. ให้นักศึกษาเขียนโปรแกรมเพื่อแปลงเลขฐานสองเป็นฐานสิบ โดยโปรแกรมมีการรับค่าเลขฐานสองจำนวน 4 หลักในรูปของเลขฐานสิบ มีรูปแบบการทำงานตามตัวอย่างด้านล่าง

```
Enter binary number : 1010
Decimal value of 1010 = 10
```

6. ให้นักศึกษาเขียนโปรแกรมเพื่อคำนวณหาความเร็วเฉลี่ยของรถยนต์ โดยโปรแกรมมีการรับค่ากิโลเมตรเริ่มต้น ค่ากิโลเมตรสิ้นสุด และจำนวนเวลาที่ใช้ไป มีรูปแบบการทำงานตามตัวอย่างด้านล่าง

<u>สูตร</u> ความเร็วเฉลี่ย = ระยะทาง / เวลาที่ใช้ทั้งหมด

```
Data inputs are integer!.
Enter start kilometer : 200
Enter end kilometer : 347
Enter time used(hour minute second) : 2 12 34

Car traveled 147 kilometers in 2 hrs 12 min 34 sec.
Average velocity was 66.5326 kph.
```