



259201

COMPUTER PROGRAMMING FOR ENGINEERS

Week 2

Variables, Constant, and Operators

Objectives

- โครงสร้างของ C++ โปรแกรม (program structure)
- ตัวแปร (variables)
 - ชนิดของข้อมูล (data types)
 - การประกาศตัวแปร (variable declaration)
- ค่าคงที่ (constant)
- ตัวดำเนินการ (operators)

Structure of a C++ Program

```
1 // This is my first C++ program
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     int age;
7
8     cout << "Hello World!!! ";
9     cout << "How old are you?: ";
10    cin >> age;
11
12    // Display output
13    cout << "So... you are " << age << "years old?\n";
14    cout << "Congratulations to your..." << endl;
15    cout << "FIRST C++ PROGRAM..." << "\t good luck.";
16
17    return 0;
18 }
```

Comments

Header file

Main function declaration

Main function definition
ประกอบไปด้วยชุดคำสั่ง
(expression) ซึ่งกำหนด
การทำงานให้โปรแกรม

ระบุว่าโปรแกรมสามารถทำงาน
จนกระทั่งจบโปรแกรมได้

ทุกคำสั่ง (expression) ต้องถูกปิดท้ายด้วยเครื่องหมาย **semicolon** `;`

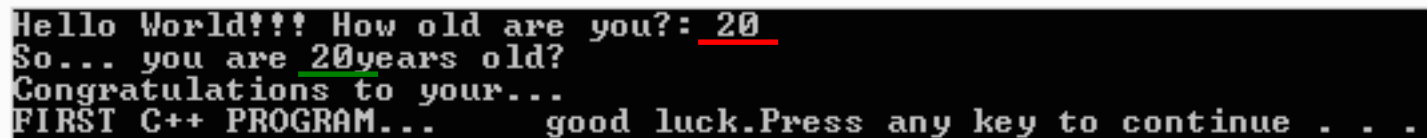
How does a variable looks like?

```
1 // This is my first C++ program
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     int age;
7
8     cout << "Hello World!!! ";
9     cout << "How old are you?: ";
10    cin >> age;
11
12    // Display output
13    cout << "So... you are " << age << "years old?\n";
14    cout << "Congratulations to your..." << endl;
15    cout << "FIRST C++ PROGRAM..." << "\t good luck.";
16
17
18 }
```

1. ประกาศตัวแปรก่อนใช้งาน

2. กำหนดค่าให้ตัวแปร

3. ดึงค่าของตัวแปรมาใช้



```
Hello World!!! How old are you?: 20
So... you are 20years old?
Congratulations to your...
FIRST C++ PROGRAM... good luck.Press any key to continue . . .
```

ช่วยให้โปรแกรมสามารถจดจำข้อมูลเพื่อนำไปใช้ในภายหลังได้!!!

Fundamental Data Types

- C++ เป็นภาษาที่ต้องประกาศตัวแปรก่อนจะนำตัวแปรไปใช้งานเสมอ และในการประกาศตัวแปรต้องระบุชนิดของข้อมูลที่จะจัดเก็บด้วย
- ชนิดของตัวแปร สามารถแบ่งได้ตามชนิดของข้อมูลที่จะจัดเก็บ ได้แก่
 - **Character types** - ข้อมูลอักขระ หรือการเก็บข้อมูลเป็นตัวอักษร 1 ตัว
 - `'A'`, `'a'`, `'5'`, `'.'`, `'_'`, `' '` (space)
 - `'\t'` (tab), `'\n'` (newline)
 - **หมายเหตุ:** `'5'` ไม่สามารถนำไปใช้ในการคำนวณทางคณิตศาสตร์ได้
 - **Numerical Integer types** - เลขจำนวนเต็มที่สามารถนำไปใช้คำนวณได้
 - 7, 1024, 65535, -127, 0
 - 7u (unsigned int), 7l (long), 7ul (unsigned long)
 - 0x4b, 0xFF (เลขฐาน 16 หรือ Hexadecimal)
 - 0113, 0720 (เลขฐาน 8)

Fundamental Data Types

- ชนิดของตัวแปร (ต่อ)
 - **Floating-point types** - เลขทศนิยม หรือจำนวนจริง ใช้ในการคำนวณได้
 - Default type - `double`
 - `3.14159`, `0.01`
 - `6.02e23`, `1.75e-9`
 - `3.0`
 - `3.14159L` (long double)
 - `6.02e23f` (float)
 - **Boolean type** - ค่าความจริง ซึ่งมักใช้ในการทดสอบเงื่อนไขต่างๆ
 - `false` (0)
 - `true` (any values other than 0)

Fundamental Data Types

Group	Type names*	Notes on size / precision
Character types	<code>char</code>	Exactly one byte in size. At least 8 bits.
	<code>char16_t</code>	Not smaller than <code>char</code> . At least 16 bits.
	<code>char32_t</code>	Not smaller than <code>char16_t</code> . At least 32 bits.
	<code>wchar_t</code>	Can represent the largest supported character set.
Integer types (signed)	<code>signed char</code>	Same size as <code>char</code> . At least 8 bits.
	<code>signed short int</code>	Not smaller than <code>char</code> . At least 16 bits.
	<code>signed int</code>	Not smaller than <code>short</code> . At least 16 bits.
	<code>signed long int</code>	Not smaller than <code>int</code> . At least 32 bits.
	<code>signed long long int</code>	Not smaller than <code>long</code> . At least 64 bits.
Integer types (unsigned)	<code>unsigned char</code>	(same size as their signed counterparts)
	<code>unsigned short int</code>	
	<code>unsigned int</code>	
	<code>unsigned long int</code>	
	<code>unsigned long long int</code>	
Floating-point types	<code>float</code>	
	<code>double</code>	Precision not less than <code>float</code>
	<code>long double</code>	Precision not less than <code>double</code>
Boolean type	<code>bool</code>	

Size	Unique representable values	Notes
8-bit	256	$= 2^8$
16-bit	65 536	$= 2^{16}$
32-bit	4 294 967 296	$= 2^{32}$ (~4 billion)
64-bit	18 446 744 073 309 551 616	$= 2^{64}$ (~18 billion billion)

Declaration of variables

- การประกาศตัวแปร (Declaration)

```
type identifier_list;
```

```
int a;  
float mynumber;
```

```
int a, b, c;
```

```
int a;  
int b;  
int c;
```

- เป็นการจองพื้นที่ในหน่วยความจำหลักของคอมพิวเตอร์ (RAM) เพื่อใช้เก็บข้อมูลชนิดที่ได้ระบุไว้
 - มีการตั้งชื่อให้กับหน่วยความจำตำแหน่งนั้นๆ เพื่อความสะดวกในการอ้างอิง
- การกำหนดค่าเริ่มต้นให้ตัวแปร (Initialization)

```
type identifier = initial_value;
```

```
int a = 1, b = 5;
```

```
char charp = '#', at = '@';
```


Example

- ตัวอย่าง - ผลลัพธ์ที่แสดงออกทางหน้าจอคือ?

เกิดอะไรขึ้นในหน่วยความจำ?

```
1 // operating with variables
2
3 #include <iostream>
4 using namespace std;
5
6 int main ()
7 {
8     // declaring variables:
9     int a, b;
10    int result;
11
12    // process:
13    a = 5;
14    b = 2;
15    a = a + 1;
16    result = a - b;
17
18    // print out the result:
19    cout << result;
20
21    // terminate the program:
22    return 0;
23 }
```

a		int a, b;
b		
result		int result;
a	5	a = 5;
b	2	b = 2;
result		
a	6	a = a+1;
b	2	
result	4	result = a-b;

Declaration of variables

- ชื่อตัวแปร (identifier) – *CASE SENSITIVE*
 - ประกอบด้วยตัวอักษร ตัวเลข หรือเครื่องหมาย ‘_’ (underscore)
 - ไม่ขึ้นต้นด้วยตัวเลข ไม่มี “__” (double underscore) อยู่ในชื่อ
 - ไม่ซ้ำกับคำสำคัญ (reserved keyword) ที่ C++ ได้จองไว้ใช้งาน

`alignas, alignof, and, and_eq, asm, auto, bitand, bitor, bool, break, case, catch, char, char16_t, char32_t, class, compl, const, constexpr, const_cast, continue, decltype, default, delete, do, double, dynamic_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, noexcept, not, not_eq, nullptr, operator, or, or_eq, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_assert, static_cast, struct, switch, template, this, thread_local, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t, while, xor, xor_eq`

IDE และ Compiler จะแจ้งให้เราทราบเมื่อมีการตั้งชื่อไม่ถูกต้อง

Declaration of variables

- ตัวอย่างการตั้งชื่อตัวแปร
 - `first_name` **valid**
 - `last-name` **invalid** – contains '-'
 - `include` **invalid** – match a keyword
 - `YearOne` **valid**
 - `40days` **invalid** – starts with numeric character
 - `example#1` **invalid** – contains '#'
 - `_long` **valid**
 - `goto` **invalid** – match a keyword
 - `float__a` **invalid** – contains double underscores
 - `using` **invalid** – match a keyword

String

- **String** - จัดอยู่ในกลุ่มของตัวแปรประกอบ (compound type) ซึ่งจัดเก็บข้อมูลชนิดอักขระ (char) หลายตัวเข้าด้วยกัน
 - ใช้จัดเก็บ คำ และข้อความ
- ต้อง `#include <string>` ในส่วน preprocessor

```
// my first string
#include <iostream>
#include <string>
using namespace std;

int main ()
{
    string mystring;
    mystring = "This is a string";
    cout << mystring;
    return 0;
}
```

This is a string

String

- เราสามารถใช้งานตัวแปรชนิด **String** นี้ได้แบบเดียวกับตัวแปรชนิดอื่นๆ
 - การกำหนดหรือไม่กำหนดค่าเริ่มต้น
 - การเปลี่ยนแปลงค่าระหว่างการทำงานของโปรแกรม

```
#include <iostream>
#include <string>
using namespace std;
```

```
int main ()
{
    string mystring;
    mystring = "This is the initial string content";
    cout << mystring << endl;
    mystring = "Now... It's different string\n";
    cout << mystring;
    mystring = "How to display '\\\\' and '\\\"' ";
    cout << mystring;
    return 0;
}
```

```
This is the initial string content
Now... It's different string
How to display '\\' and '\"'
```

Special Characters

- อักขระพิเศษ (escape code)
 - อักขระซึ่งยากที่ระบุลงในโค้ดของโปรแกรม ปกติแล้วจะนำหน้าด้วย escape character (\) เช่น การขึ้นบรรทัดใหม่ การแท็บ การลบอักขระ

Escape code	Description
\n	newline
\r	carriage return
\t	tab
\v	vertical tab
\b	backspace
\f	form feed (page feed)
\a	alert (beep)
\'	single quote (')
\"	double quote (")
\?	question mark (?)
\\	backslash (\)

Typed Constant Expression

- การกำหนดค่าคงที่
 - หน่วยความจำบริเวณที่ใช้เก็บค่าคงที่ จะไม่อนุญาตให้แก้ไขข้อมูลได้
 - โดยทั่วไปจะต้องระบุชนิดของข้อมูล (data type) ด้วยเสมอ

`const type identifier = value;`

```
#include <iostream>
using namespace std;

const double pi = 3.14159;
const char newline = '\n';

int main ()
{
    double r=5.0;           // radius
    double circle;

    circle = 2 * pi * r;
    cout << circle;
    cout << newline;
}
```

31.4159

Operators

- ตัวดำเนินการ (operator)
 - เราสามารถนำตัวแปร (variable) และค่าคงที่ (constant) มาใช้ในการคำนวณหรือประมวลผลต่างๆได้โดยอาศัยตัวดำเนินการต่างๆ
- Assignment Operator (=)
 - ใช้ในการกำหนดค่าให้กับตัวแปร โดยคัดลอกค่าจากด้านขวา (right-hand side)
ไปยังตัวแปรที่อยู่ด้านซ้าย (left-hand side) ของเครื่องหมาย =

```
int main ()
{
    int a, b;           // a:?, b:?
    a = 10;             // a:10, b:?
    b = 4;              // a:10, b:4
    a = b;              // a:4, b:4
    b = 7;              // a:4, b:7

    cout << "a:";
    cout << a;
    cout << " b:";
    cout << b;
}
```

```
y = 2 + (x = 5);
```

```
x = 5;
y = 2 + x;
```

```
x = y = z = 5;
```


Operators

- **Arithmetic Operators** (+, -, *, /, %)

- ตัวดำเนินการทางเลขคณิต
- **Modulo (%)** – คำนวณหาเศษของการหาร
 - $10 \% 2 = 0$ $11 \% 2 = 1$
 - $8 \% 6 = 2$ $2 \% 6 = 2$

operator	description
+	addition
-	subtraction
*	multiplication
/	division
%	modulo

- **Compound Assignment** (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

```
#include <iostream>
using namespace std;
```

```
int main ()
{
    int a=5, b=3;
    a+=2;  a*=b;
    b/=3;  b%=2;
    cout << "a: " << a;
    cout << ", b: " << b;
}
```

a: 21, b: 1

expression	equivalent to...
$y += x;$	$y = y + x;$
$x -= 5;$	$x = x - 5;$
$x /= y;$	$x = x / y;$
$price *= units + 1;$	$price = price * (units+1);$

ใช้เมื่อต้องการย่อคำสั่งที่ใช้ operator ทางคณิตศาสตร์ที่กำหนดให้สั้นลง

Operators

- **Increment และ Decrement Operators** (++ , --)

- คำสั่งทางคณิตศาสตร์บางคำสั่งสามารถลดรูปให้สั้นลงได้อีก
- Increment operator (++) ใช้เพิ่มค่าให้กับตัวแปรขึ้นทีละ 1 เหมือนกับ +=1
- Decrement operator (--) ใช้ลดค่าตัวแปรลงทีละ 1 เหมือนกับ -=1
- ดังนั้น ++x; มีค่าเหมือนกับ x+=1; และ x=x+1;
- สามารถวาง operator ไว้ได้ทั้งก่อนหน้า (prefix) หรือด้านหลัง (suffix) ตัวแปร
 - y=++x; หรือ y=--x; (prefix) : เพิ่มหรือลดค่า x ก่อนกำหนดค่า x ให้ y
 - y=x++; หรือ y=x--; (suffix) : เพิ่มหรือลดค่า x หลังกำหนดค่า x ให้ y

Example 1	Example 2
<pre>x = 3; y = ++x; // x contains 4, y contains 4</pre>	<pre>x = 3; y = x++; // x contains 4, y contains 3</pre>

Operators

- **Relational** และ **comparison operators** (`==`, `!=`, `>`, `<`, `>=`, `<=`)
 - ใช้ในการเปรียบเทียบค่าสองค่า โดยจะให้ผลลัพธ์เป็น `true` หรือ `false`
 - ข้อสังเกต: `=` คือการกำหนดค่า (assign) ในขณะที่ `==` ใช้ในการเปรียบเทียบ

operator	description
<code>==</code>	Equal to
<code>!=</code>	Not equal to
<code><</code>	Less than
<code>></code>	Greater than
<code><=</code>	Less than or equal to
<code>>=</code>	Greater than or equal to

```
(7 == 5)    // evaluates to false
(5 > 4)     // evaluates to true
(3 != 2)    // evaluates to true
(6 >= 6)    // evaluates to true
(5 < 5)     // evaluates to false
```

- หากกำหนดให้ `a=2`, `b=3`, `c=6`

```
(a == 5)    // evaluates to false, since a is not equal to 5
(a*b >= c)  // evaluates to true, since (2*3 >= 6) is true
(b+4 > a*c) // evaluates to false, since (3+4 > 2*6) is false
((b=2) == a) // evaluates to true
```

Operators

- Logical Operators (!, &&, ||)

- ใช้สำหรับการคำนวณทางตรรกะ โดยจะให้ผลลัพธ์เป็น `true` หรือ `false`
- `!` ใช้ในการหาค่า **นิเสธ** (NOT) หรือ **ค่าตรงข้าม**

```
!(5 == 5)    // evaluates to false because the expression at its right (5 == 5) is true
!(6 <= 4)    // evaluates to true because (6 <= 4) would be false
!true        // evaluates to false
!false       // evaluates to true
```

- `&&` และ `||` ใช้คำนวณหาค่าทางตรรกะของการ **and** และ **or** ตามลำดับ

&& OPERATOR (and)		
a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

OPERATOR (or)		
a	b	a b
true	true	true
true	false	true
false	true	true
false	false	false

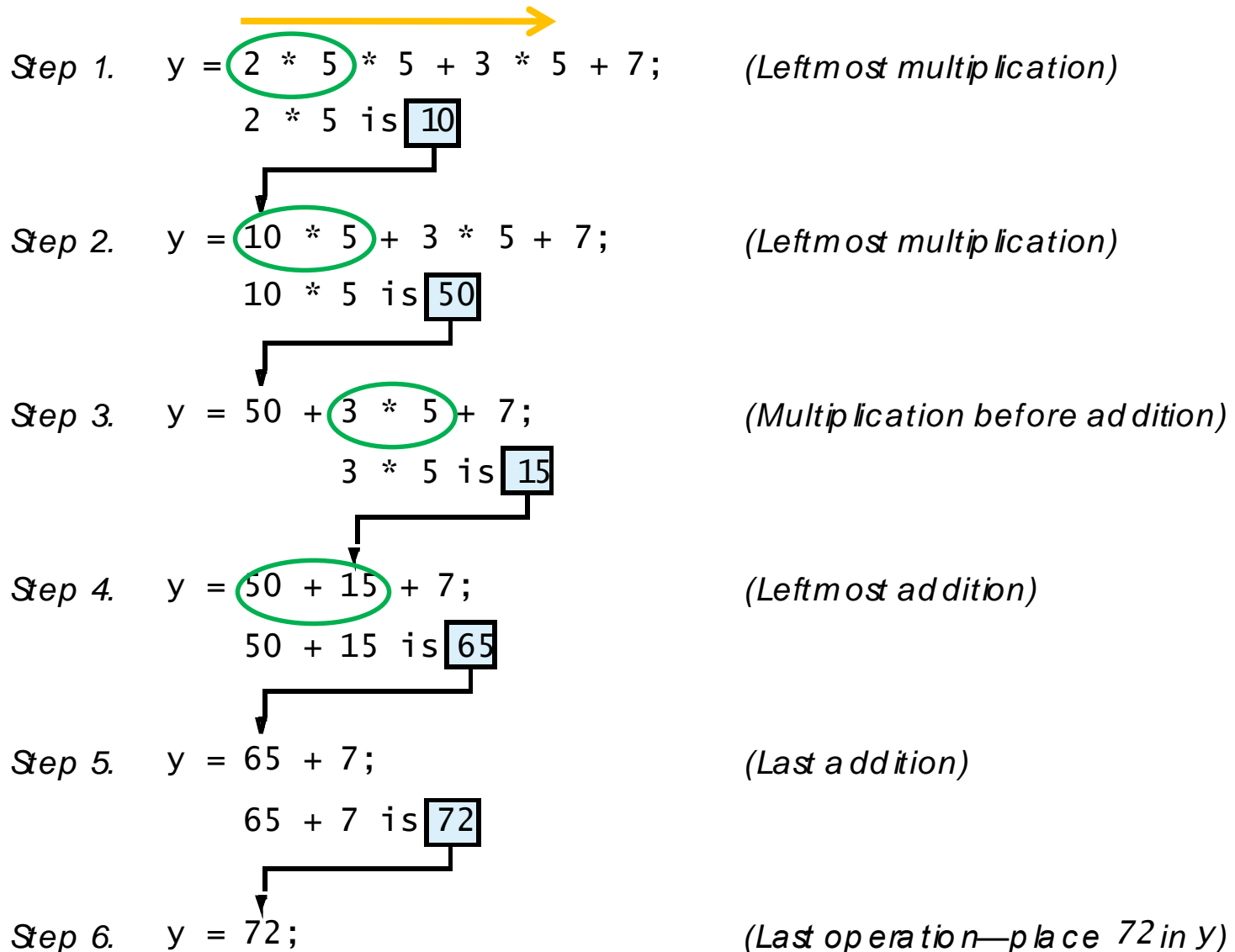
```
( (5 == 5) && (3 > 6) ) // evaluates to false ( true && false )
( (5 == 5) || (3 > 6) ) // evaluates to true ( true || false )
```

Precedence of Operators

- การจัดลำดับการทำงานของตัวดำเนินการ

ลำดับความสำคัญ	เครื่องหมาย	การดำเนินการ
1	()	Left to Right
2	!, ++, --, (typecast)	<u>Right to Left</u>
3	*, /, %	Left to Right
4	+, -	Left to Right
5	<, <=, >, >=	Left to Right
6	=, !=	Left to Right
7	&&	Left to Right
8		Left to Right
9	*=, /=, %=, +=, -=	Left to Right

Example



Mathematical Operations with C++

- เป็นการยากที่จะทำการคำนวณทางคณิตศาสตร์ที่ซับซ้อน โดยอาศัยเพียงตัวดำเนินการทางคณิตศาสตร์ตามที่กล่าวมาแล้ว
- C++ ได้เตรียมชุดคำสั่ง (หรือฟังก์ชัน) ในการคำนวณพื้นฐานที่จำเป็นทางคณิตศาสตร์ไว้ให้ผู้พัฒนาโปรแกรมได้เรียกใช้งาน
 - คำสั่งเหล่านี้ถูกจัดเตรียมไว้อยู่ใน Header file ที่ชื่อว่า **cmath**

```
#include <cmath>
```

- กลุ่มของชุดคำสั่งพื้นฐานทางคณิตศาสตร์ที่ C++ มีให้เรียกใช้งาน
 - **Trigonometric functions** – sin, cos, tan, asin, acos, ...
 - **Exponential and Logarithmic functions** – exp, log, log2, log10, ...
 - **Power functions** – pow, sqrt, ...
 - **Rounding functions** – ceil, floor
 - **Other functions** – abs, ...

Examples

```
1  #include <iostream>  /* cin, cout */
2  #include <cmath>     /* math operations */
3  using namespace std;
4
5  int main ()
6  {
7      double input, output;
8      input = 1024;
9      output = sqrt(input);
10     cout << "sqrt(" << input << ") = " << output << endl;
11     cout << "11 ^ 3 = " << pow(11, 3) << endl;
12     cout << "e ^ 2 = " << exp(2) << endl;
13     cout << "log2(16) = " << log2(16) << endl;
14     cout << "log(10000) = " << log(10000) << endl;
15     cout << "log10(10000) = " << log10(10000) << endl;
16
17     const double PI = 3.141592;
18     input = 30;
19     output = sin(input*PI/180);
20     cout << "sine 30 radian = " << sin(30) << endl;
21     cout << "sine 30 degree = " << output << endl;
22
23     return 0;
24 }
```

```
sqrt(1024) = 32
11 ^ 3 = 1331
e ^ 2 = 7.38906
log2(16) = 4
log(10000) = 9.21034
log10(10000) = 4
sine 30 radian = -0.988032
sine 30 degree = 0.5
```


Summary

- **ตัวแปร** (variable) ถูกนำมาใช้ในการช่วยจดจำข้อมูลเพื่อใช้งานในภายหลัง ต้องอาศัยการจองพื้นที่ในหน่วยความจำ และมีการอ้างอิงโดยใช้ชื่อ
- **ชนิดของตัวแปร** คือชนิดของข้อมูลที่จะถูกจัดเก็บไว้ในตัวแปร ได้แก่
 - อักขระ (Character)
 - เลขจำนวนเต็ม (Numerical Integer)
 - เลขทศนิยม (Floating-point)
 - ค่าทางตรรกะ (Boolean)
- **ค่าคงที่** (constant) สามารถพิจารณาได้เหมือนตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าในภายหลังได้ อาศัยคำสำคัญ **constant** ในการประกาศ
- การตั้งชื่อตัวแปรและค่าคงที่ ต้องไม่ซ้ำกับ**คำสำคัญ**ที่ได้มีการกำหนดไว้
- ชื่อตัวแปรและค่าคงที่ใน C++ นั้นเป็นแบบ **case sensitive**

Summary

- ตัวดำเนินการ (operators) คือเครื่องหมายที่ถูกใช้เพื่อสั่งให้โปรแกรมประมวลผล มีหลายชนิด ได้แก่
 - กำหนดค่า (Assignment)
 - คำนวณทางเลขคณิต (Arithmetic)
 - คำนวณและกำหนดค่า (Compound Assignment)
 - เพิ่มค่าและลดค่า (Increment and Decrement)
 - ความสัมพันธ์ และเปรียบเทียบ (Relational and Comparison)
 - การคำนวณทางตรรกะ (Logical)
- เมื่อภายในคำสั่งประกอบไปด้วยตัวดำเนินการมากกว่า 1 ตัว ลำดับการประมวลผลตัวดำเนินการ (precedence) ก่อนหลังจะเป็นไปตามที่ได้มีการกำหนดไว้

Summary

- การคำนวณทางคณิตศาสตร์ที่ซับซ้อนมากขึ้นสามารถทำได้โดยอาศัยชุดคำสั่ง (หรือฟังก์ชัน) ที่ C++ ได้เตรียมไว้ให้เรียกใช้
 - ต้อง `#include <cmath>` จึงจะเรียกใช้งานคำสั่งทางคณิตศาสตร์เหล่านั้นได้
 - ค่าที่ป้อนให้กับชุดคำสั่ง (input) และผลลัพธ์จากการคำนวณ (output) จะต้องเป็นเลขทศนิยม - `float`, `double`, ...
 - C++ จะแปลงค่าที่ป้อนชุดคำสั่งเป็นค่าทศนิยมให้หากจำเป็น

```
double output = sqrt(99);
```

```
//convert 99 to double (99.0)
```

- หากนำตัวแปรชนิด `int` มาเก็บค่าผลลัพธ์การคำนวณ C++ จะทำการแปลงค่าผลลัพธ์ที่ได้จากชุดคำสั่งซึ่งเป็นทศนิยม ให้กลายเป็นจำนวนเต็มโดยการปัดค่าหลังจุดทศนิยมทิ้ง (rounding)

```
int output = sqrt(99);
```

```
// output = sqrt(82) = 9
```

Labs

- Lab 2.1: ทดลองเขียนโปรแกรมดังที่กำหนดให้

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int a=20; b=30;
6      int c = 4, output;
7
8      output = a-b/c;
9      cout << "output: " << output << endl;
10 }
```

- **Compile** ผ่านหรือไม่?
- ผิดพลาดจุดใด แก้ไขอย่างไร
- ทำงานได้ถูกต้องอย่างที่เราจะเป็นหรือไม่ แก้ไขอย่างไร

Labs

- Lab 2.2: ทดลองเขียนโปรแกรมดังเพื่อคำนวณตามสมการ

$$y = 3x^3 + 2e + 2^{(2x+1)} + \sqrt{x^2 + 1}$$

กำหนดให้: **e** เป็นค่าคงที่ของ **Euler**

- **Input**: ค่า x โดยรับผ่านคีย์บอร์ดจากผู้ใช้
- **Output**: ค่า y ที่คำนวณได้จากสมการ
- **Test cases**:

```
Enter x = 2  
Result y = 63.6726
```

```
Enter x = 4  
Result y = 713.56
```

Labs

- Lab 2.3: เขียนโปรแกรมคำนวณการเคลื่อนที่ในแนวราบ

- ตัวแปรที่เกี่ยวข้อง:

- ความเร็วต้น (u) หน่วยเป็น เมตร/วินาที
- ความเร็วปลาย (v) หน่วยเป็น เมตร/วินาที
- อัตราเร่งในการเคลื่อนที่ (a) หน่วยเป็น เมตร/วินาที²
- ระยะทางในการเคลื่อนที่ (s) หน่วยเป็น เมตร
- เวลาในการเคลื่อนที่ (t) หน่วยเป็น วินาที

- **กรณีที่ 1** เคลื่อนที่จากหยุดนิ่ง ด้วยอัตราเร่ง และระยะเวลาที่กำหนด

- คำนวณหาความเร็วปลาย (v) และระยะทางการเคลื่อนที่ (s)

- **กรณีที่ 2** เคลื่อนที่ด้วยความเร็วคงที่ ลดความเร็วด้วยอัตราเร่งคงที่ ได้ระยะทางการเคลื่อนที่ภายในระยะเวลาที่กำหนด

- คำนวณหาความเร็วต้น (u) และความเร็วปลาย (v) หลังจากลดความเร็วแล้ว