SBTVC

PROGRARM C++

MR. JATURON NIYARUT

INTRODUCTION

การนำภาษา C++ ไปใช้ในการทำงาน

ทำไมถึงต้อง C++

1. ประสิทธิภาพของภาษา
2. Cross-Platform
3. การจัดการหน่วยความจำ
4. Object-Oriented Programming (OOP)
5. ใช้ในการพัฒนาซอฟแวร์ขนาดใหญ่

1. ประสิทธิภาพของภาษา

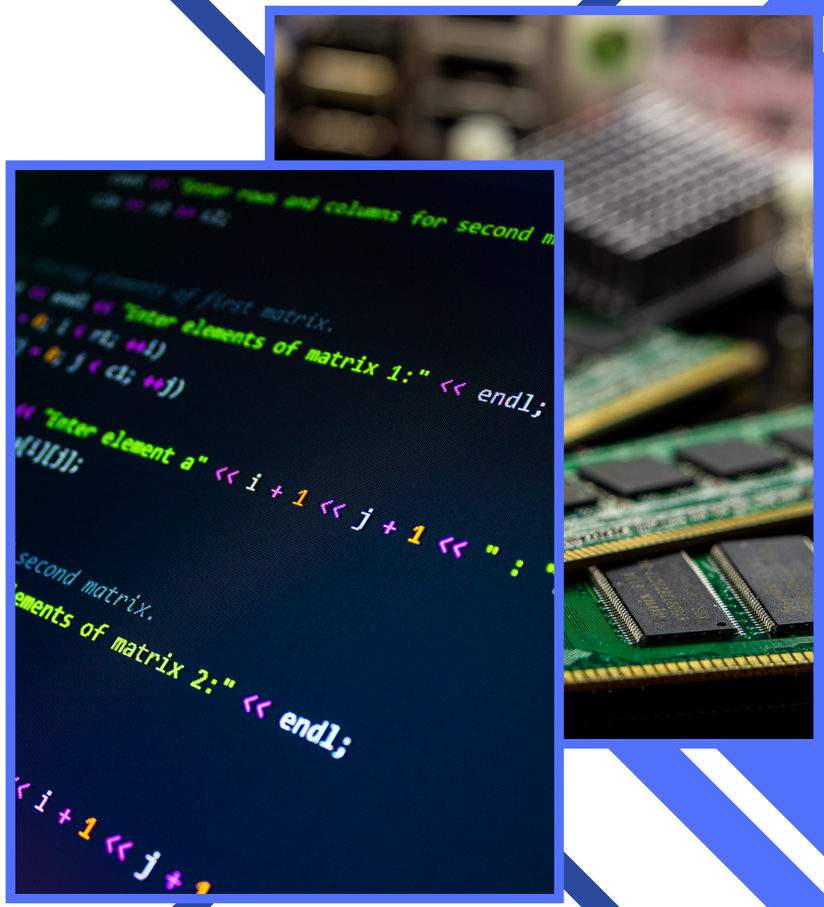
C++ ถือเป็นภาษาที่มีประสิทธิภาพสูงที่สุดในกลุ่มของภาษาที่ทำงานใกล้ชิดกับภาษาเครื่อง (LOW-LEVEL LANGUAGE) ทำให้มีการควบคุมทรัพยากรเครื่องคอมพิวเตอร์ได้มาก เหมาะสำหรับการพัฒนาโปรแกรมที่ต้องการประสิทธิภาพสูง เช่น ซอฟต์แวร์ที่ทำงานในระบบภาคเอกชน (EMBEDDED SYSTEMS) หรือแอปพลิเคชันที่ต้องการการประมวลผลที่เร็ว

2. CROSS-PLATFORM

ภาษา C++ สามารถทำงานได้บนหลายแพลตฟอร์มโดยทั่วไปโดยไม่ต้องทำการแก้ไขโค้ดมากมาย นั่นหมายความว่า, คุณสามารถพัฒนาซอฟต์แวร์ที่ทำงานทั้งบน WINDOWS, MACOS, LINUX ได้

3. การจัดการหน่วยความจำ

C++ เป็นหนึ่งในความสามารถที่สำคัญของภาษานี้ เนื่องจากมีความยืดหยุ่นและมีความเป็นระบบที่สูง ทำให้นักพัฒนาสามารถควบคุมและจัดการหน่วยความจำได้อย่างละเอียดตามความต้องการของโปรแกรมที่กำลังพัฒนาได้



4. OBJECT-ORIENTED PROGRAMMING (OOP)

C++ เป็นภาษาที่รองรับ OOP ซึ่งช่วยให้โค้ดมีความสมบูรณ์และมีความยืดหยุ่นในการพัฒนา ยกตัวอย่าง เช่น ENCAPSULATION, INHERITANCE, POLYMORPHISM, ABSTRACTION

5. ใช้ในการพัฒนาซอฟแวร์ขนาดใหญ่

หลายผลิตภัณฑ์ซอฟต์แวร์ที่ใหญ่มีการใช้ C++ เช่น MICROSOFT WINDOWS, ADOBE PHOTOSHOP, MICROSOFT OFFICE, MYSQL, และมีการใช้ C++ ในส่วนของ GAME DEVELOPMENT ที่ต้องการความประสิทธิภาพสูง

1. ระบบปฏิบัติการ: WINDOWS เป็นตัวอย่างหนึ่งของระบบปฏิบัติการที่ถูกพัฒนาด้วย C++
2. เกม: หลายเกมในอุตสาหกรรมการเกมมีการพัฒนาด้วย C++ เนื่องจากความเร็วและประสิทธิภาพของภาษา
3. ซอฟต์แวร์ฐานข้อมูล: MYSQL เป็นตัวอย่างของซอฟต์แวร์ฐานข้อมูลที่ใช้ C++ ในการพัฒนา

5. ใช้ในการพัฒนาซอฟแวร์ขนาดใหญ่

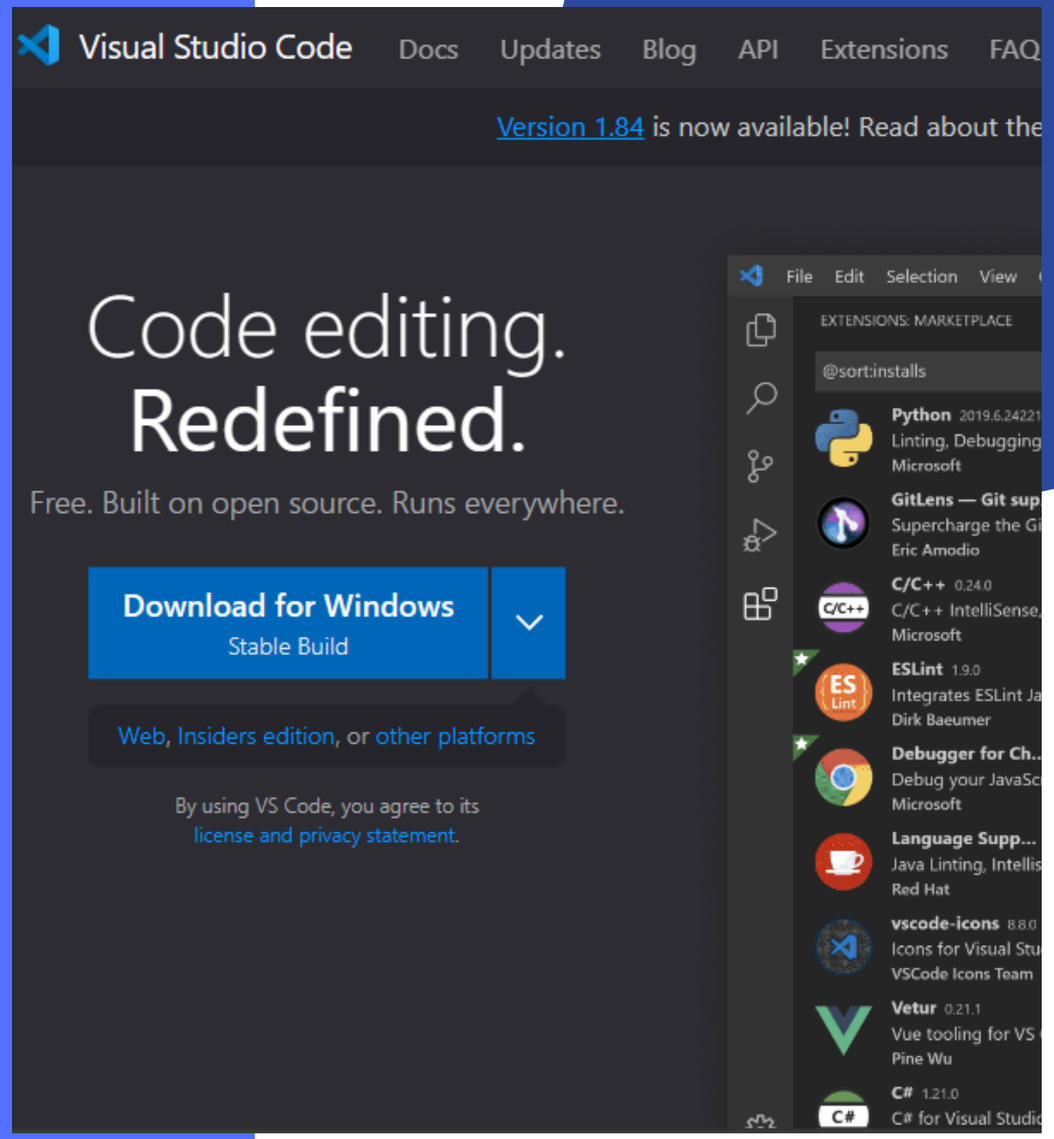
4. การควบคุมเครื่องจักร: ทำงานในอุตสาหกรรม

4.1 ระบบควบคุมโปรแกรม: C++ ถูกใช้ในการพัฒนาระบบควบคุมโปรแกรมในโรงงาน เช่น ระบบควบคุมการผลิต, ระบบการควบคุมอุปกรณ์ทางไฟฟ้า, ระบบ PLC (PROGRAMMABLE LOGIC CONTROLLER

4.2 ระบบควบคุมการเคลื่อนที่ (MOTION CONTROL): C++ ถูกนำมาใช้ในการพัฒนาระบบควบคุมการเคลื่อนที่ของหุ่นยนต์, เครื่องจักร, หรือระบบการเคลื่อนที่อื่น ๆ

HOW TO INSTALL

CODE C++ IN VISUAL STUDIO CODE



1

DOWNLOAD VSCODE

Link for download and install
https://code.visualstudio.com/

2

DOWNLOAD MINGW

Link for download and install
https://sourceforge.net/projects/mingw/

3

SET PATH AND DOWNLOAD EXTENSTION

Set Path in Windows and Download extention in Vscode and Vscode icon

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK ☘

- Introduction C++
- Why is C++
- Install MinGw
- Install Visual Studio Code
- Background of C++

SBTVC

VARIABLES
DATA TYPES

WHAT IS VARIABLES AND DATA TYPES IN C++

DATA TYPES

1. Boolean (bool)
2. Character (char)
3. Integer (int)
4. Floating point (float)
5. Double floating point (double)

DATA TYPES

6 String (string)

7 Valueless (void)

8 Wide character (wchar_t)

1

BOOLEAN

ชนิดตัวแปรแบบ *Boolean (bool)* สามารถเก็บค่าได้เฉพาะ *true* หรือ *false* มีขนาด *8 bit* หรือ *1 byte*

ตัวอย่าง *code*

```
bool statusOne = false;
bool statusTwo = true;
const bool statusThree = false;
const bool statusFour = true;
```

2

CHARACTER

ชนิดตัวแปรแบบ *Character (char)* สามารถเก็บค่าตัวอักษรเพียงตัวเดียว มีขนาด *8 bit* หรือ *1 byte*

ตัวอย่าง *code*

```
char charOne = ‘A’;
char charTwo = ‘B’;
const char charThree = ‘0’;
const char charFour = ‘$’;
```

3

INTEGER

ชนิดตัวแปรแบบ *Integer (int)* สามารถเก็บค่าจำนวนเต็มบวกหรือลบ มีขนาด *32 bit* หรือ *4 byte*

ตัวอย่าง *code*

```
int numberOne = 0;
int numberTwo = 999999999;
const int numberThree = -123;
const int numberFour = -15479;
```

4

FLOATING

ชนิดตัวแปรแบบ *Floating point (float)* สามารถเก็บค่าทศนิยมได้สูงสุด มีขนาด *32 bit* หรือ *4 byte*

ตัวอย่าง *code*

```
float numberOne = 0.00;
float numberTwo = 99.9999999;
const float numberThree = -130.123;
const float numberFour = -15.479;
```

5

DOUBLE

ชนิดตัวแปรแบบ *Floating point (float)* สามารถเก็บค่าทศนิยมได้สูงสุด มีขนาด *64 bit* หรือ *8 byte*

ตัวอย่าง *code*

```
double numberOne = 0.00;
double numberTwo = 99.9999999;
const double numberThree = -130.123;
const double numberFour = -15.479;
```

6

STRING

ชนิดตัวแปรแบบ *String (string)* สามารถเก็บค่าเป็นตัวหนังสือที่ขนาดหลายตัวอักษรขนาดของ *string* ขึ้นอยู่กับข้อความว่ามีกี่ตัวอักษร

ตัวอย่าง *code*

```
string nameOne = “Hello world.”;
string nameTwo = “99.9999999”;
const string nameThree = “!@#qwedqwe”;
const string nameFour = “”;
```

7

VALUELESS

ชนิดตัวแปรแบบ *Valueless (void)* จะไม่สามารถเก็บค่าได้แต่จะถูกใช้เป็นการประกาศ *function* หรือ เป็นประเภทของ *Pointer* ที่ไม่ได้ระบุประเภทชี้ไป

ตัวอย่าง *code*

```
void myFunction();
void myFunction(int numberOne);
void myFunction(string name)
```

8

WIDE

ชนิดตัวแปรแบบ *Wide character (wchar_t)* จะถูกใช้ในส่วนของการแทนที่โดยตรงใน *ASCII* เช่น *‘A’* จะมีค่าเท่ากับ *65*

ตัวอย่าง *code*

```
wchar_t wideCharOne = L'A'; // 65
wchar_t wideCharTwo = L'B'; // 66
const wchar_t wideCharThree = L'C'; // 67
const wchar_t wideCharFour = L'D'; // 68
```

EXERCISE

1. จงประกาศตัวแปรที่มีชนิด Boolean **(bool)** มาจำนวน 10 ตัวแปร และ แสดงค่า
2. จงประกาศตัวแปรที่มีชนิด Character **(char)** มาจำนวน 10 ตัวแปร และ แสดงค่า
3. จงประกาศตัวแปรที่มีชนิด Integer **(int)** มาจำนวน 10 ตัวแปร และ แสดงค่า
4. จงประกาศตัวแปรที่มีชนิด Floating point **(float)** มาจำนวน 10 ตัวแปร และ แสดงค่า
5. จงประกาศตัวแปรที่มีชนิด Floating point **(dobule)** มาจำนวน 10 ตัวแปร และ แสดงค่า
6. จงประกาศตัวแปรที่มีชนิด String **(string)** มาจำนวน 10 ตัวแปร และ แสดงค่า
7. จงประกาศตัว function ที่มีชนิด Valueless **(void)** มาจำนวน 10 function และ แสดงค่า “A”
8. จงประกาศตัวแปรที่มีชนิด Wide character **(wchar_t)** มาจำนวน 10 ตัวแปร และ แสดงค่า

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK ☘

- Data Types in C++
- Example **Data Types in C++**
- **Exercise with C++**

SBTVC

CONDITIONAL OPERATORS

CONDITIONAL STATEMENTS AND OPERRATORS IN C++

OPERATORS

1 **เครื่องหมายบวก (+)**

```
int a = 10;
int b = 12;
int sum = a + b; // 22
```

2 **เครื่องหมายลบ (-)**

```
double valueA = 12.14534;
double valueB = 28.684455;
double sum = valueA - valueB; // 40.829795
```

3 **เครื่องหมายคูณ (*)**

```
float valueA = 3.521;
float valueB = 15.98;
float sum = valueA * valueB; // 56.26558
```

OPERATORS

4 **เครื่องหมายหาร (/)**

```
double valueA = 233.623;
double valueB = 11.234;
double sum = valueA / valueB; // 20.796066
```

5 **เครื่องหมาย MODULUS (%)**

```
int a = 10;
int b = 3;
int sum = a % b; // 1
```

STATEMENTS

1 **IF (เงื่อนไขหลัก)**

```
if (true or false) {
    // ถ้าหากเงื่อนไขเป็นจริงจะทำงานภายใต้ปีกกานี้
}
```

2 **ELSE IF (เงื่อนไขรอง)**

```
else if (true or false) {
    // ถ้าหากเงื่อนไขเป็นจริงจะทำงานภายใต้ปีกกานี้
}
```

3 **ELSE (ไม่เข้าเงื่อนไขทั้งหมด)**

```
else {
    // ถ้าหากไม่เข้าในเงื่อนไขของ if() และ else if()
   // จะทำงานภายใต้ปีกกานี้
}
```

STATEMENTS

1

SWITCH CASE

ในการใช้ *switch case* สามารถใช้ได้เฉพาะเงื่อนไขที่เป็นตัวเลขเท่านั้นจะแตกต่างกับ *if else* ที่สามารถใช้งานได้ยืดยุ่นมากกว่า

```
int choice = 1;

 switch (choice) {
   case 1:
     std::cout << "You chose One. \n";
     break;

   case 2:
     std::cout << "You chose Two. \n";
     break;

   default:
     std::cout << "Invalid choice. \n";
     break;
 }
```

CONDITIONS

1 **เครื่องหมายน้อยกว่า (a < b)**

```
int a = 10;
int b = 20;
bool status = (a < b); // true or 1
```

2 **เครื่องหมายมากกว่า (a > b)**

```
int a = 30;
int b = 20;
bool status = (a > b); // true or 1
```

3 **เครื่องหมายน้อยกว่าหรือเท่ากับ (a <= b)**

```
int a = 40;
int b = 40;
bool status = (a <= b); // true or 1
```

CONDITIONS

4 **เครื่องหมายมากกว่าหรือเท่ากับ (a >= b)**

```
int a = 51;
int b = 50;
bool status = (a >= b); // true or 1
```

5 **เครื่องหมายเท่ากับ (a == b)**

```
int a = 100;
int b = 100;
bool status = (a == b); // true or 1
```

6 **เครื่องหมายไม่เท่ากับ (a != b)**

```
int a = 888;
int b = 999;
bool status = (a != b); // true or 1
```

CONDITIONS

7 **เครื่องหมาย logical AND (&&)**

```
int a = 1;
int b = 0;
bool status = (a && b); // false or 0
```

8 **เครื่องหมาย logical OR (||)**

```
int a = 1;
int b = 1;
bool status = (a || b); // true or 1
```

9 **เครื่องหมาย NOT (!)**

```
int a = 0;
int b = 1;
bool status = !(a && b); // true or 1
```

EXERCISE

1. ประกาศตัวแปรชนิดตัวแปรตามที่เห็น ตัวแปรที่หนึ่ง ตัวแปรที่สอง แล้วนำตัวแปรทั้งสองตัวมาคำนวณกันตามเครื่องหมายที่กำหนดและแสดงผลลัพท์ออกมา

1.1 ตัวแปรที่ 1 มีค่าเท่ากับ 554
ตัวแปรที่ 2 มีค่าเท่ากับ 987
เครื่องหมาย +

1.2 ตัวแปรที่ 1 มีค่าเท่ากับ 123
ตัวแปรที่ 2 มีค่าเท่ากับ 4,123
เครื่องหมาย -

1.3 ตัวแปรที่ 1 มีค่าเท่ากับ 1,478
ตัวแปรที่ 2 มีค่าเท่ากับ 9,874
เครื่องหมาย *

1.4 ตัวแปรที่ 1 มีค่าเท่ากับ 26,988
ตัวแปรที่ 2 มีค่าเท่ากับ 19,874
เครื่องหมาย +

EXERCISE

1.5 ตัวแปรที่ 1 มีค่าเท่ากับ 5.54
ตัวแปรที่ 2 มีค่าเท่ากับ 9.87
เครื่องหมาย /

1.6 ตัวแปรที่ 1 มีค่าเท่ากับ 41.2367
ตัวแปรที่ 2 มีค่าเท่ากับ 44,123.45
เครื่องหมาย +

1.7 ตัวแปรที่ 1 มีค่าเท่ากับ 51,478.87
ตัวแปรที่ 2 มีค่าเท่ากับ 59,874.75
เครื่องหมาย -

1.8 ตัวแปรที่ 1 มีค่าเท่ากับ 26,988.51
ตัวแปรที่ 2 มีค่าเท่ากับ 19,874.7885
เครื่องหมาย *

1.9 ตัวแปรที่ 1 มีค่าเท่ากับ 54,126,988.6557
ตัวแปรที่ 2 มีค่าเท่ากับ 19,879,874.5678
เครื่องหมาย /

EXERCISE

1.10 ตัวแปรที่ 1 มีค่าเท่ากับ 987
ตัวแปรที่ 2 มีค่าเท่ากับ 13
เครื่องหมาย %

1.11 ตัวแปรที่ 1 มีค่าเท่ากับ 125
ตัวแปรที่ 2 มีค่าเท่ากับ 9
เครื่องหมาย %

1.12 ตัวแปรที่ 1 มีค่าเท่ากับ 51,478.87
ตัวแปรที่ 2 มีค่าเท่ากับ 59,874.75
เครื่องหมาย *

1.13 ตัวแปรที่ 1 มีค่าเท่ากับ 26,988.51
ตัวแปรที่ 2 มีค่าเท่ากับ 19,874.7885
เครื่องหมาย -

1.14 ตัวแปรที่ 1 มีค่าเท่ากับ 54,126,988.6557
ตัวแปรที่ 2 มีค่าเท่ากับ 19,879,874.5678
เครื่องหมาย /

EXERCISE

1.15 ตัวแปรที่ 1 มีค่าเท่ากับ 9827
ตัวแปรที่ 2 มีค่าเท่ากับ 133
เครื่องหมาย *

1.16 ตัวแปรที่ 1 มีค่าเท่ากับ 1215
ตัวแปรที่ 2 มีค่าเท่ากับ 92
เครื่องหมาย *

1.17 ตัวแปรที่ 1 มีค่าเท่ากับ 51,478.87
ตัวแปรที่ 2 มีค่าเท่ากับ 59,874.75
เครื่องหมาย -

1.18 ตัวแปรที่ 1 มีค่าเท่ากับ 2,988.51
ตัวแปรที่ 2 มีค่าเท่ากับ 19,874.7885
เครื่องหมาย -

1.19 ตัวแปรที่ 1 มีค่าเท่ากับ 1,126,988.6557
ตัวแปรที่ 2 มีค่าเท่ากับ 19,879,874.5678
เครื่องหมาย /

EXERCISE

2. นำค่าผลจากข้อที่กำหนดมาเปรียบเทียบตามเครื่องหมาย (Conditions) ที่กำหนด

2.1 ผลลัพท์ข้อที่ 1.1 <= 1.2
2.2 ผลลัพท์ข้อที่ 1.3 == 1.9
2.3 ผลลัพท์ข้อที่ !(1.9 != 1.7)
2.4 ผลลัพท์ข้อที่ 1.10 >= 1.8
2.5 ผลลัพท์ข้อที่ 1.12 >= 1.14
2.6 ผลลัพท์ข้อที่ !(1.17 <= 1.15)
2.7 ผลลัพท์ข้อที่ 1.18 == 1.8
2.8 ผลลัพท์ข้อที่ !(1.19 != 1.6)
2.9 ผลลัพท์ข้อที่ 1.7 > 1.14
2.10 ผลลัพท์ข้อที่ 1.8 < 1.16
2.11 ผลลัพท์ข้อที่ 1.9 <= 1.7
2.12 ผลลัพท์ข้อที่ !(1.5 >= 1.4)
2.13 ผลลัพท์ข้อที่ 1.7 == 1.6

EXERCISE

2.14 ผลลัพท์ข้อที่ 1.12 <= 1.1
2.15 ผลลัพท์ข้อที่ 1.18 == 1.2
2.16 ผลลัพท์ข้อที่ !(!(1.19 != 1.4))
2.17 ผลลัพท์ข้อที่ 1.16 >= 1.9
2.18 ผลลัพท์ข้อที่ 1.9 >= 1.6
2.19 ผลลัพท์ข้อที่ !(1.11 <= 1.4)
2.20 ผลลัพท์ข้อที่ 1.12 == 1.9
2.21 ผลลัพท์ข้อที่ 1.13 != 1.8
2.22 ผลลัพท์ข้อที่ !(1.17 > 1.7)
2.23 ผลลัพท์ข้อที่ 1.18 < 1.6
2.24 ผลลัพท์ข้อที่ !(1.14 <= 1.17)
2.25 ผลลัพท์ข้อที่ 1.15 >= 1.14
2.26 ผลลัพท์ข้อที่ 1.17 == 1.16
2.27 ผลลัพท์ข้อที่ !(1.18 <= 1.2)
2.28 ผลลัพท์ข้อที่ 1.17 != 1.4
2.29 ผลลัพท์ข้อที่ 1.14 < 1.5

EXERCISE

3. นำค่าผลจากข้อที่กำหนดมาเปรียบเทียบตามเครื่องหมาย (Statements) ที่กำหนด

```
3.1 if ( ข้อที่ 2.22 ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 1
    } else if ( ข้อที่ 2.14 ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 2
    } else if ( ข้อที่ 2.19 ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 3
    } else if ( !(ข้อที่ 2.24) ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 4
    } else {
       // แสดงผลคำว่า ไม่เข้าเงื่อนไขทั้ง 4
    }
```

EXERCISE

```
3.2 if ( ข้อที่ 2.12 ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 1
    } else if ( ข้อที่ 2.28 ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 2
    } else {
      // แสดงผลคำว่า ไม่เข้าเงื่อนไขทั้ง 2
    }
```

```
3.2 if ( ข้อที่ 2.1 ) {
      // แสดงผลคำว่า เข้าเงื่อนไขที่ 1
    } else if ( ข้อที่ 2.23 ) {
      // แสดงผลคำว่า เข้าเงื่อนไขที่ 2
    } else {
      // แสดงผลคำว่า ไม่เข้าเงื่อนไขทั้ง 2
    }
```

EXERCISE

3.4 if (ข้อที่ 2.12) {
// แสดงผลคำว่า เข้าเงื่อนไขที่ 1
} else if (ข้อที่ 2.28) {
// แสดงผลคำว่า เข้าเงื่อนไขที่ 2
} else {
// แสดงผลคำว่า ไม่เข้าเงื่อนไขทั้ง 2
}

3.5 if (ข้อที่ 2.1) {
// แสดงผลคำว่า เข้าเงื่อนไขที่ 1
} else if (ข้อที่ 2.23) {
// แสดงผลคำว่า เข้าเงื่อนไขที่ 2
} else {
// แสดงผลคำว่า ไม่เข้าเงื่อนไขทั้ง 2
}

EXERCISE

```
3.6 if ( ข้อที่ 2.18 ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 1
    }  else {
       // แสดงผลคำว่า ไม่เข้าเงื่อนไขที่ 1
    }

3.7 if ( ข้อที่ 2.1 ) {
       // แสดงผลคำว่า เข้าเงื่อนไขที่ 1
    } else if ( ข้อที่ 2.23 ) {
       // แสดงผลคำว่า เข้าเงื่อนไขที่ 2
    }  else if ( ข้อที่ 2.12 ) {
       // แสดงผลคำว่า เข้าเงื่อนไขที่ 3
    }  else {
       // แสดงผลคำว่า ไม่เข้าเงื่อนไขทั้ง 3
    }
```

EXERCISE

```
3.8 if ( !(ข้อที่ 2.14 && ข้อที่ 2.18) ) {
        // แสดงผลคำว่า เข้าเงื่อนไขที่ 1
    }  else {
       // แสดงผลคำว่า ไม่เข้าเงื่อนไขที่ 1
    }

3.9 if ( ข้อที่ 2.23 || ข้อที่ 2.11 ) {
       // แสดงผลคำว่า เข้าเงื่อนไขที่ 1
    } else if ( ข้อที่ 2.23 && ข้อที่ 2.4 ) {
       // แสดงผลคำว่า เข้าเงื่อนไขที่ 2
    }  else if ( ข้อที่ 2.12 && ข้อที่ 2.9 ) {
       // แสดงผลคำว่า เข้าเงื่อนไขที่ 3
    }  else {
       // แสดงผลคำว่า ไม่เข้าเงื่อนไขทั้ง 3
    }
```

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK ☘

- Operators in C++
- Statements in C++
- Conditions in C++
- Logical AND Logical AND
- Exercise with C++

SBTVC

FOR LOOP REPETITION

ABOUT FOR LOOP IN C++

1

FOR LOOP

การใช้ for loop จะแนะนำให้ใช้ต่อเมื่อเรารู้จำนวนรอบที่แน่นอนในการวนรอบเช่น ตั้งการให้วน 10 รอบในการทำงานประมวลผล

ตัวอย่าง *code*

```
for (int = 0 ; i < 10 ; i++) {
   //  เมื่อใช้คำสั่ง cout << i << “ ”; จะแสดงผล
  //   0 1 2 3 4 5 6 7 8 9
}
```

2

WHILE LOOP

การใช้ While loop จะแนะนำให้ใช้ต่อเมื่อเราไม่รู้จำนวนรอบที่แน่นอนของการวนรอบว่าต้องมีการวนรอบกี่ครั้งเพื่อทำการประมวลผล

ตัวอย่าง *code*

```
int value = 15641621654651;
while (value > 0) {
   value = value - 18;
   // เมื่อใช้คำสั่ง cout << value << " "; จะแสดงผล
   // 27 24 21 18 15 12 9 6 3 0
}
```

3

DO WHILE LOOP

การใช้ Do while loop จะคล้ายกับของ While loop แตกต่างกันคือ While loop จะเช็คเงื่อนไขก่อน ทำงานส่วนของ Do while จะทำก่อนแล้วเช็คเงื่อนไข

ตัวอย่าง *code*

```
int value = 30;
do {
    value = value -3;
    // เมื่อใช้คำสั่ง cout << value << “ ”; จะแสดงผล
    // 27 24 21 18 15 12 9 6 3 0
}
while (value > 0)
```

EXAMPLE

การใช้ FOR LOOP เบื้องต้น

```
for (int i = 0 ; i < 10 ; i++) {
    cout << i << " ";
}
```

OUTPUT

0 1 2 3 4 5 6 7 8 9

```
for (int i = 0 ; i <= 10 ; i++) {
    cout << i << " ";
}
```

OUTPUT

0 1 2 3 4 5 6 7 8 9 10

```
for (int i = 10 ; i >= 0 ; i--) {
    cout << i << " ";
}
```

OUTPUT

10 9 8 7 6 5 4 3 2 1 0

EXAMPLE

สามารถใช้ LOOP ซ้อน LOOP ได้

```
for (int i = 0 ; i < 12 ; i++) {
    cout << i << ": ";

    for (int j = 0 ; j < 10 ; j++) {
        count << j << " ";
    }

    cout << "\n";
}
```

OUTPUT

```
0: 0 1 2 3 4 5 6 7 8 9
1: 0 1 2 3 4 5 6 7 8 9
2: 0 1 2 3 4 5 6 7 8 9
3: 0 1 2 3 4 5 6 7 8 9
4: 0 1 2 3 4 5 6 7 8 9
5: 0 1 2 3 4 5 6 7 8 9
6: 0 1 2 3 4 5 6 7 8 9
7: 0 1 2 3 4 5 6 7 8 9
8: 0 1 2 3 4 5 6 7 8 9
9: 0 1 2 3 4 5 6 7 8 9
10: 0 1 2 3 4 5 6 7 8 9
11: 0 1 2 3 4 5 6 7 8 9
```

EXAMPLE

สามารถใช้ LOOP ซ้อน LOOP ได้

```
for (int i = 0 ; i < 12 ; i++) {
    cout << i << “: ”;

    for (int j = 0 ; j < i ; j++) {
        count << j << “ ”;
    }

    cout << “\n”;
}
```

OUTPUT

```
0:
1: 0
2: 0 1
3: 0 1 2
4: 0 1 2 3
5: 0 1 2 3 4
6: 0 1 2 3 4 5
7: 0 1 2 3 4 5 6
8: 0 1 2 3 4 5 6 7
9: 0 1 2 3 4 5 6 7 8
10: 0 1 2 3 4 5 6 7 8 9
11: 0 1 2 3 4 5 6 7 8 10
```

EXAMPLE

สามารถใช้ LOOP ซ้อน LOOP ได้

```
for (int i = 0 ; i < 12 ; i++) {
    cout << i << “: ”;

    for (int j = i ; i > 0 ; j--) {
        count << j << “ ”;
    }

    cout << “\n”;
}
```

OUTPUT

```
0:
1: 1
2: 2 1
3: 3 2 1
4: 4 3 2 1
5: 5 4 3 2 1
6: 6 5 4 3 2 1
7: 7 6 4 3 2 1
8: 8 7 6 5 4 3 2 1
9: 9 8 7 6 5 4 3 2 1
10: 10 9 8 7 6 5 4 3 2 1
11: 11 10 9 8 7 6 5 4 3 2 1
```

EXAMPLE

การใช้ WHILE LOOP เบื้องต้น

```
int value = 30;
while (value > 10) {
   cout << value << “ ”;
   value -= 5;
}
```

OUTPUT

30 25 20 15

```
int value = 0;
while (value < 200) {
   value += 38;
   cout << value << “ ”;
}
```

OUTPUT

38 76 114 152 190 228

EXAMPLE

การใช้ WHILE LOOP เบื้องต้น

```
int value = 30;
do {
    cout << value << “ ”;
    value -= 5;
}
while (value > 10)
```

OUTPUT

30 25 20 15

```
int value = 0;
do {
    cout << value << “ ”;
    value += 38;
}
while (value < 2000)
```

OUTPUT

0 38 76 114 152 190

EXERCISE

1.1 จงเขียนโปรแกรมให้แสดงดังผลลัพธ์ต่อไปนี้ โดยใช้ For while และ Do while

```
********************
********************
****************
**************
************
**********
**********
*********
********
*******
******
*****
****
***
**
*
```

EXERCISE

1.2 จงเขียนโปรแกรมให้แสดงดังผลลัพธ์ต่อไปนี้ โดยใช้ For while และ Do while

```
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
****************
*****************
******************
```

EXERCISE

1.3 จงเขียนโปรแกรมให้แสดงดังผลลัพธ์ต่อไปนี้ โดยใช้ For while และ Do while

```
*
**
***
****
*****
******
*******
********
*********
*********
********
*******
******
*****
****
***
**
*
```

EXERCISE

1.4 จงเขียนโปรแกรมให้แสดงดังผลลัพธ์ต่อไปนี้ โดยใช้ For while และ Do while

```
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
```

EXERCISE

1.5 จงเขียนโปรแกรมแม่สูตรคูณให้ผลลัพธ์ต่อไปนี้ โดยใช้ For while และ Do while และต้องสามารถรองรับได้ทั้งหมดในแม่สูตรคูณ

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
2 x 11 = 22
2 x 12 = 24

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK 🍀

- For loop in C++
- While loop in C++
- Do while loop in C++
- Exercise with C++

SBTVC

ARRAY
1D 2D AND 3D

ABOUT ARRAY 1D, 2D AND 3D IN C++

1

ARRAY 1D

array 1 มิติ คือโครงสร้างข้อมูลที่เก็บข้อมูลในลักษณะที่เป็นชุดของข้อมูลที่มีค่าเรียงติดกันไปในทิศทางเดียวกัน หรือ ชนิดตัวตัวแปรชนิดเดียวกันที่เก็บไว้ในตัวแปรเดียวกัน *Index* ของ *array* เริ่มจากตำแหน่งที่ *0*

array 1	array 2	array 3	array 4
0	1	2	3
สมพงษ์	เหรียญเงิน	87	A
สมชาย	ทองคำ	84	A
สมหญิง	เงินแท้	75	B
สมศรี	ทองเปลว	68	C+
สมศักดิ์	ทองก้อน	49	F

ARRAY 1D

	0	1	2	3	4	5	6	7
Type Int	64	154	975	-105	67	34	58	-100

array ตำแหน่งที่ 0 จะมีค่าเท่ากับ 64

array ตำแหน่งที่ 3 จะมีค่าเท่ากับ -105

array ตำแหน่งที่ 6 จะมีค่าเท่ากับ 58

array ตำแหน่งที่ 1 จะมีค่าเท่ากับ 154

ADDRESS

PHYSICAL ADDRESS

Physical address เป็นที่อยู่จริงในอุปกรณ์ในหน่วยความจำของคอมพิวเตอร์บน hardware ของคอมพิวเตอร์

ตัวอย่างเช่น
ถ้าหน่วยความจำ (RAM) มีขนาด 32GB บนความพิวเตอร์ ที่อยู่ที่เป็น physical address จะบอกตำแหน่งที่แท้จริงเช่น 0x00000000, 0x00000001, 0x00000002 ตามลำดับไปเรื่อยๆ

ADDRESS

LOGICAL ADDRESS

Logical address เป็นที่อยู่ที่ระบบปฏิบัติการหรือแอปพลิเคชันใช้เพื่ออ้างอิงถึงข้อมูลในหน่วนความจำหรือทรัพยากรต่างๆ ผู้ใช้งานไม่จำเป็นต้องเข้าถึงหน่วยความจำ (RAM) โดยเราจะเรียกใช้จากตัวแปรที่สร้างขึ้นมา

ตัวอย่างเช่น

ถ้าประกาศตัวแปร int A = 12; สิ่งที่จะเกิดขึ้นก็คือ OS จะไปหาพื้นที่ว่างใน หน่วยความจำ (RAM) แล้วนำ ค่า 12 ไปเก็บไว้ใน Physical address แล้วจะทำการ return Physical address มาเก็บไว้ในตัวแปร A เมื่อมีการอ้างถึงตัวแปร A ก็จะนำ Physical address ไปเรียกจากข้อมูลในหน่วยความจำ (RAM)

ARRAY 1D

	0	1	2	3	4	5	6	7
Type Int	64	154	975	-105	67	34	58	-100

```
int a = array[0];  // 64

int b = array[3];  // -105

int c = array[6];  // 58

int d = array[1];  // 154
```

Code:

```
int arrayNumber[8] = {64, 154, 975, -105, 67, 34, 58, -100};
```

ARRAY 1D

	0	1	2	3	4	5	6	7
Type Int	64	154	975	-105	67	34	58	-100

```
int sumA = array[0] + array[4];  // 64 + 67 = 131

int sumB = array[2] + array[7];  // 975 + (-100) = 875

int sumC = array[6] + array[5];  // 58 + 34 = 102

int sumD = array[4] + array[1];  // 67 + 154 = 221
```

Code:

```
int arrayNumber[8] = {64, 154, 975, -105, 67, 34, 58, -100};
```

ARRAY 1D

	0	1	2	3	4	5	6	7
Type string	“PLC”	“KBTG”	“SCB”	“KYC”	“AWS”	“API”	“MQTT”	“IoT”

array ตำแหน่งที่ 0 จะมีค่าเท่ากับ “PLC”

array ตำแหน่งที่ 3 จะมีค่าเท่ากับ “KYC”

array ตำแหน่งที่ 6 จะมีค่าเท่ากับ “MQTT”

array ตำแหน่งที่ 1 จะมีค่าเท่ากับ “KBTG”

Code:

```
string arrayText[8] = {"PLC", "KBTG", "SCB", "KYC", "AWS", "API",
"MQTT", "IoT"};
```

ARRAY 1D

```
int array[8] = {64, 154, 975, -105, 67, 34, 58, -100};

for (int i = 0 ; i < 8 ; i++) {
    cout << array[i] << " ";
}
```

OUTPUT 64 154 975 -105 67 34 58 -100

```
for (int i = 0 ; i < 8 ; i++) {
    double x = array[i] + 10;
    cout << x << " ";
}
```

OUTPUT 74 164 985 -95 77 44 68 -90

ARRAY 1D

```
int array[8] = {64, 154, 975, -105, 67, 34, 58, -100};

for (int i = 0 ; i < 7 ; i++) {
      int x = array[i];
      int y = array[i + 1];
      int sum = x + y;
      array[i] = sum;
      cout <<  array[i] << " ";
}
```

OUTPUT

218 1129 870 -38 101 92 -42

ARRAY 1D

```
int array[8] = {64, 154, 975, -105, 67, 34, 58, -100};

for (int i = 7 ; i > 0 ; i--) {
      int x = array[i];
      int y = array[i - 1];
      int sum = x - y;
      array[i] = sum;
      cout <<  array[i] << " ";
}
```

OUTPUT

```
-158 24 -33 172 -1080 821 90
```

2

ARRAY 2D

array 2 มิติ คือโครงสร้างข้อมูลที่เก็บข้อมูลในลักษณะที่เป็นชุดของข้อมูลที่มีค่าเรียงติดกันไปในทิศทางเดียวกัน หรือ ชนิดตัวตัวแปรชนิดเดียวกันที่เก็บไว้ในตัวแปรเดียวกัน Index ของ array เริ่มจากตำแหน่งที่ 0, 0 โดยจะใช้วิธีการหาจำนวนช่องของ array ก็คือ x และ y
สามารถนำไปใช้ในงานที่เกี่ยวข้องกับรูปภาพ รวมไปถึง Image processing ได้อีกด้วย

ARRAY 2D

Type Int

	0	1	2	3
0	64	154	975	-105
1	123	451	234	541
2	845	54	98	44
3	92	123	456	909
4	763	862	97	42

Size array [5][4] (5x4 ช่อง)

```
int a = array[0][0];  // 64
int b = array[0][1];  // 154
int c = array[0][2];  // 975
int d = array[1][3];  // 541
int e = array[4][1];  // 763
int f = array[2][2];  // 98
int g = array[3][3];  // 909
```

ARRAY 2D

Type Int

	0	1	2	3	4
0	12	135	-878	879	654
1	785	678	-787	465	785
2	467	985	-56	123	642
3	987	745	87	813	83
4	65	678	45	645	98

Size array [5][5] (5x5 ช่อง)

```
int a = array[4][4];
int b = array[3][4];
int c = array[2][3];
int d = array[1][3];
int e = array[2][3];
int f = array[3][1];
int g = array[4][3];
```

ARRAY 2D

Type Int

	0	1	2	3	4
0	54	145	-105	512	56
1	723	456	787	758	78
2	451	578	56	456	97
3	541	785	-87	478	-75
4	265	862	845	125	-87

Size array [5][5] (5x5 ช่อง)

```
int a = array[2][3];
int b = array[1][4];
int c = array[4][1];
int d = array[3][2];
int e = array[2][3];
int f = array[1][4];
int g = array[3][3];
```

ARRAY 2D

Type Int

	0	1	2	3	4
0	-54	15	-5	12	556
1	23	46	7	78	578
2	41	58	76	476	197
3	51	78	-77	488	-11
4	26	82	75	165	-78

Size array [5][5] (5x5 ช่อง)

```
int a = array[2][3];
int b = array[1][4];
int sum = a / b;


int c = array[3][2];
int d = array[2][3];
int sum = ( c * d ) - 1000;
```

ARRAY 2D

Type Int

	0	1	2	3	4
0	54	145	-105	512	56
1	723	456	787	758	78
2	451	578	56	456	97
3	541	785	-87	478	-75
4	265	862	845	125	-87

Size array [5][5] (5x5 ช่อง)

```
int array[5][5] = {
    {54, 145, -105, 512, 56},
    {723, 456, 787, 758, 78},
    {451, 578, 56, 456, 97},
    {541, 785, -87, 478, -75},
    {265, 862, 845, 125, -87},
};
```

ARRAY 2D

การหาผลคูณเมทริกซ์

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

หมายเหตุ: ขนาดแถวแนวตั้ง (row) ของ array ที่หนึ่งต้องมีค่าเท่ากับ ขนาดแถวแนวนอน (col) ของ array ที่สอง เท่านั้น

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

2		4		5				
X		X		X				
7		1		4				
=		=		=				
14	+	4	+	20				

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38			

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

2		4		5				
X		X		X				
8		2		5				
=		=		=				
16	+	8	+	25				

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38	49		

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

2		4		5				
X		X		X				
0		3		1				
=		=		=				
0	+	12	+	5				

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38	49	17	

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

2		4		5				
X		X		X				
9		3		6				
=		=		=				
18	+	12	+	30				

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38	49	15	60

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

5		1		2
X		X		X
7		1		4
=		=		=
35	+	1	+	8

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38	49	17	60
44			

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

5		1		2
X		X		X
8		2		5
=		=		=
40	+	2	+	10

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38	49	17	60
44	52		

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

5		1		2
X		X		X
0		3		1
=		=		=
0	+	3	+	2

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38	49	17	60
44	52	5	

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

หมายเหตุ: ขนาดใหม่ของ เมทริกซ์จะจำนวนช่องเท่ากับ col ของ array ที่หนึ่ง และ row ของ array ที่สอง คูณกัน 2x4

5		1		2
X		X		X
9		3		6
=		=		=
45	+	3	+	12

	0	1	2
0	2	4	5
1	5	1	2

Size: 2x3

X

	0	1	2	3
0	7	8	0	9
1	1	2	3	3
2	4	5	1	6

Size: 3x4

=

38	49	17	60
44	52	5	60

Size: 2x4

ARRAY 2D

การหาผลคูณเมทริกซ์

```
int arrayA[2][3] = {
    {2, 4, 5},
    {5, 1, 2}
};

int arrayB[3][4] = {
    {7, 8, 0, 9},
    {1, 2, 3, 3},
    {4, 5, 1, 6}
};

int arraySum[2][4];
```

```
for (int i = 0 ; i < 2 ; i++) {

    for (int j = 0 ; j < 4 ; j++) {

      int sum = 0;
      for (int x = 0 ; x < 3 ; x++) {
        sum += arrayA[i][x] * arrayB[x][j];
      }

      arraySum[i][j] = sum;
      cout << arraySum[i][j] << ", ";
    }
    cout << "\n";
  }
```

OUTPUT

```
38, 49, 17, 60,
44, 52, 5, 60,
```

ARRAY 2D

Type string

	0	1	2	3
0	“Ab9”	“Ab8”	“Ab6”	“Ab4”
1	“Bc1”	“Bc2”	“Bc9”	“Bc7”
2	“Cz7”	“Cz9”	“Cz2”	“Cz4”
3	“Ex9”	“Ex1”	“Ex3”	“Ex6”
4	“Fo7”	“Fo8”	“Fo6”	“Fo5”

Size array [5][4] (5x4 ช่อง)

```
int a = array[0][0];  // “Ab9”
int b = array[0][1];  // “Ab8”
int c = array[0][2];  // “Ab6”
int d = array[1][3];  // “Bc7”
int e = array[4][1];  // “Fo8”
int f = array[2][2];  // “Cz2”
int g = array[3][3];  // “Ex6”
```

3

ARRAY 3D

array 3 มิติ คือโครงสร้างข้อมูลที่เก็บข้อมูลในลักษณะที่เป็นชุดของข้อมูลที่มีค่าเรียงติดกันไปในทิศทางเดียวกัน หรือ ชนิดตัวตัวแปรชนิดเดียวกันที่เก็บไว้ในตัวแปรเดียวกัน Index ของ array เริ่มจากตำแหน่งที่ 0, 0, 0 โดยจะใช้วิธีการหาจำนวนช่องของ array ก็คือ x, y, z
สามารถนำไปใช้ในงานที่เกี่ยวข้องกับรูปภาพ 3D รวมไปถึง Image processing แบบ 3D ได้อีกด้วย

EXERCISE

1.1 จงสร้าง array 1D ให้มีขนาด 10 ช่องและเพิ่มข้อมูล array แต่ละช่องโดยให้มีค่าเท่ากับ 13, 26, 39, 52, 65, 78, 91, 104, 117, 130 โดยใช้ for loop ในการเพิ่มค่าแต่ละช่องใน array

1.2 จงสร้าง array 1D ให้มีขนาด 10 ช่องและเพิ่มข้อมูล array แต่ละช่องโดยให้มีค่าเท่ากับ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

1.3 จงสร้าง array 1D ให้มีขนาด 10 ช่องและเพิ่มข้อมูล array แต่ละช่องโดยให้มีค่าเท่ากับ 7, 14, 21, 28, 35, 42, 49, 56, 63, 70

1.4 จงสร้าง array 1D ให้มีขนาด 10 ช่องนำค่าจาก array ข้อที่ 1.1 1.2 1.3 มารวมกันโดยจะได้ผลลัพธ์เท่ากับ 20, 60, 90, 120, 150, 180, 210, 240, 270, 300

EXERCISE

2.1 จงสร้าง array 2D ให้มีขนาด 8x4 และให้ทำการสุ่มค่าตัวเลขเข้าไปใน array ให้ครบทุกช่อง

2.2 จงสร้าง array 2D ให้มีขนาด 4x5 และให้ทำการสุ่มค่าตัวเลขเข้าไปใน array ให้ครบทุกช่อง

2.3 จงนำ array 2D จากข้อที่ 2.1 และ 2.2 มาคูณกันและแสดงผลลัพธ์ออกมา (8x5)

2.4 จงสร้าง array 2D ให้มีขนาด 5x10 และให้ทำการสุ่มค่าตัวเลขเข้าไปใน array ให้ครบทุกช่อง

2.5 จงนำ array 2D จากข้อที่ 2.3 และ 2.4 มาคูณกันและแสดงผลลัพธ์ออกมา (8x10)

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK 🍀

- Array 1D in C++
- Aarry 2D in C++
- Array 3D in C++
- Exercise with C++

SBTVC

POINTERS

POINTERS

ABOUT POINTERS IN C++

1

POINTERS

Pointers เป็นองค์ประกอบที่สำคัญในภาษา *C++* และมีการใช้งานที่หลากหลายในการจัดการและเข้าถึงข้อมูลในหน่วยความจำของคอมพิวเตอร์ซึ่งเป็นหลักฐาน *16* ดังนั้นนี่คือบางข้อมูลที่เกี่ยวข้องกับการใช้ *pointers* ใน *C++*

POINTERS

รูปแบบการเขียน point สามารถเขียนได้ 4 รูปแบบดังนี้ จะเป็นการเก็บ physical address และการเรียกใช้งานจะใช้เป็น logical address

```
int *p1;
int  * p2;
int* p3;
char *c3;
```

POINTERS

```
int m = 20;
int n = 7;

cout << “m” << m << “ &m = ” << &m < “\n”;
cout << “n” << n << “ &n = ” << &n < “\n”;

int *p1 = &m;

cout << “p1 = ” << p1 << “\n”;
cout << “*p1 = ” << *p1 << “\n”;
// *p1 - dereferencing or indirection
```

OUTPUT

```
m = 20 &m = 0061FF2C
n = 7 &n = 0061FF28
p1 = 0061FF2C
*p1 = 20
```

POINTERS

```
int *p1 = &m;

cout << “p1 = ” << p1 << “\n”;
cout << “*p1 = ” << *p1 << “\n”;
// *p1 - dereferencing or indirection

m = 99;
cout << “p1 = ” << p1 << “\n”;
cout << “*p1 = ” << *p1 << “\n”;
```

OUTPUT

```
p1 = 0061FF2C
*p1 = 20
p1 = 0061FF2C
*p1 = 99
```

POINTERS

```
int price = 15;
int *coke, *fanta, *sprite;
coke = &price;
fanta = &price;
sprite = &price;

cout << “price = ” << price << “\n”;
cout << “*coke = ” << *coke << “\n”;
cout << “*fanta = ” << *fanta << “\n”;
cout << “*sprite = ” << *sprite << “\n”;
```

OUTPUT

```
price = 15
code = 15
fanta = 15
sprite = 15
```

EXERCISE

1.1 จงสร้างตัวแปรที่เป็น pointer โดยกำหนดค่าหลักที่ตัวแปรที่หนึ่ง และ มีตัวแปรที่รอรับค่าของ physical address อยู่ทั้งหมดเพิ่มไปอีก 5 ตัวแปร โดยเมื่อมีการเปลี่ยนแปลงค่าตัวแปรที่หนึ่งให้ แสดงผลลัพธ์ค่าตัวแปรทั้งหมด

1.2 จงสร้างตัวที่เป็น array 2D ขนาด 4x4 ขึ้นมาสองตัวแปรและทำการสุ่มกำหนดค่าใน array ที่หนึ่งให้ครบ และใช้ array ที่สองนั้นเก็บค่า physical address และทำการแสดงผลลัพธ์ค่าใน array สอง

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK ☘

- Pointer in C++
- Exercise with C++

SBTVC

PREPROCESSORS HEADER FILES

PREPROCESSORS AND HEADER FILES

1

PREPROCESSORS

ใน *C++*, *Preprocessor* คือส่วนหนึ่งของภาษาที่ทำหน้าที่ประมวลผลก่อนที่โปรแกรมจะถูกรวม *(compile)* และมีหน้าที่หลายอย่าง เช่น การกำหนดค่าคงที่ *(define constants)*, การ *include* ไฟล์ *(include files)*, การใช้เงื่อนไข *(conditional compilation)* และอื่น ๆ ต่อไปนี้คือรายการ *Preprocessors* ที่ใช้ใน *C++*

PREPROCESSORS

1. #define: ใช้กำหนดค่าคงที่ (constants) หรือตัวแปรแบบไม่กำหนดชนิด (macros).
   Code: #define PI 3.14159

2. #include: ใช้ include ไฟล์ (header files) เข้ามาในโปรแกรม.
   Code: #include <iostream>

3. #ifdef, #ifndef, #else, #endif: ใช้สำหรับควบคุมการคอมไพล์ของโค้ดตามเงื่อนไข.
   Code:

```
#if defined(PLATFORM_WINDOWS)
    // โค้ดที่จะถูกคอมไพล์เมื่อ PLATFORM_WINDOWS ถูกกำหนด
#elif defined(PLATFORM_LINUX)
    // โค้ดที่จะถูกคอมไพล์เมื่อ PLATFORM_LINUX ถูกกำหนด
#else
    // โค้ดที่จะถูกคอมไพล์เมื่อไม่มีเงื่อนไขไหนถูกต้อง
#endif
```

PREPROCESSORS

4. #pragma: ใช้เพื่อให้คำสั่งคอมไพล์พิเศษ.
   Code: #pragma once

5. #undef: ใช้ยกเลิกการกำหนดค่าคงที่หรือตัวแปรแบบไม่กำหนดชนิด.
   Code: #undef PI

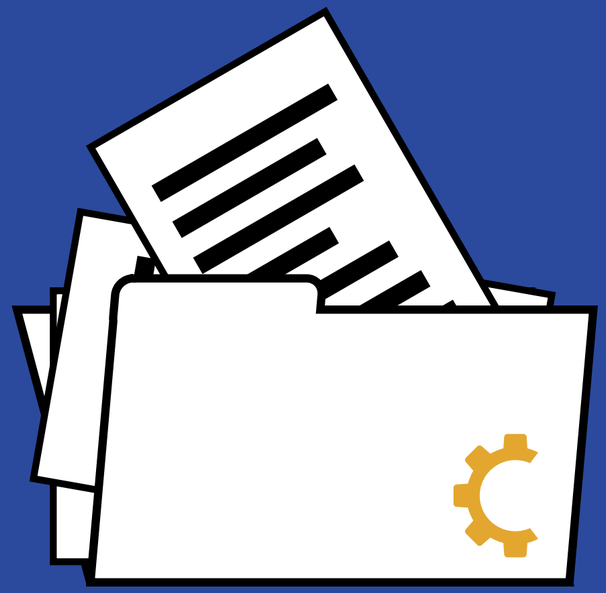
6. #error: ใช้สร้างข้อผิดพลาดในกรณีที่ตรวจสอบเงื่อนไขที่ไม่ถูกต้อง.
   Code:

```
#ifndef MY_CONSTANT
    #error "MY_CONSTANT is not defined!"
#endif
```

2

HEADER FILES

ใน *C++*, *Header Files* ถูกใช้เพื่อเก็บการประกาศ *(declarations)* ของฟังก์ชัน, คลาส, และตัวแปรที่จะถูกใช้ในหลายไฟล์ของโปรแกรม เราสามารถสร้าง *Header Files* ที่มีนามสกุล *.h* หรือ *.hpp* และแล้วใช้ *#include* เพื่อนำเข้า *(include)* ไฟล์นี้ในไฟล์ *C++* อื่น ๆ ที่ต้องการใช้งาน.



HEADER FILES

```
// myheader.h
#ifndef MYHEADER_H  // ป้องกันการ include หลายครั้ง
#define MYHEADER_H

// ประกาศฟังก์ชัน
void printMessage();

// ประกาศค่าคงที่
const int MAX_VALUE = 100;

#endif
```

HEADER FILES

```
// main.cpp
#include <iostream>
#include "myheader.h"  // นำเข้า Header File ที่เราสร้าง

int main() {
    printMessage();  // เรียกใช้ฟังก์ชันจาก Header File
    cout << "Max Value: " << MAX_VALUE;  // เรียกใช้ค่าคงที่จาก Header File

    return 0;
}
```

EXERCISE

1. จงสร้างไฟล์ header ขึ้นมาจำนวน 2 ไฟล์ให้แต่ละไฟล์นั้นมีค่าตัวแปรที่เป็น int และ string อย่างละ 1 ชนิด และทำการเรียกใช้งาน header มาใช้ที่ main file และทำการแสดงผลลัพธ์
2. จงสร้างไฟล์ header มาทั้งหมด 5 ไฟล์โดย ให้ไฟล์ที่ 1 ไปเรียกใช้งานค่าจากไฟล์ที่ 2 และไฟล์ที่ 2 ไปเรียกใช้งานค่าจากไฟล์ที่ 3 เรียงไปจนถึงไฟล์สุดท้าย และ เรียกใช้ไฟล์ที่ 1 ที่ main file และทำการแสดงผลลัพธ์

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK ☘

- Preprocessors in C++
- Header Files in C++
- Exercise with C++

SBTVC

ARGUMENTS LIBRARY

FUNCTION ARGUMENTS LIBRARY FUNCTIONS AND VARIABLE SCOPE

1

ARGUMENTS

ใน *C++*, สามารถใช้ *Function Arguments* ในหลายลักษณะต่าง ๆ ตามความต้องการของโปรแกรมของคุณ

FUNCTION ARGUMENTS

Pass by Value (ผ่านค่า)

```
void myFunction(int x) {
    // ทำสิ่งที่ต้องการกับ x
    x = x * 10;
}

int main() {
    int num = 10;
    myFunction(num);
    cout << num << “\n”;
    return 0;
}
```

OUTPUT 10 // เป็นค่าที่ไม่ได้ pass มาจาก function

FUNCTION ARGUMENTS

Pass by Reference (ผ่านอ้างอิง)

```
void myFunction(int &x) {
    // ทำสิ่งที่ต้องการกับ x
    x = x * 10;
}

int main() {
    int num = 10;
    myFunction(num);
    cout << num << “\n”;
    return 0;
}
```

OUTPUT

```
100 // เป็นค่าที่ pass มาจาก function
```

FUNCTION ARGUMENTS

Pass by Pointer (ผ่านตัวแปรประเภทพอยน์เตอร์)

```
void myFunction(int *x) {
  // ทำสิ่งที่ต้องการกับ x
  *x = *x * 10;
}

int main() {
  int num = 10;
  cout << &num << "\n";
  myFunction(&num);
  cout << num << "\n";
  return 0;
}
```

OUTPUT

```
0x61ff0c // เป็น physical address
100 // เป็นค่าที่ pass มาจาก physical address
```

FUNCTION ARGUMENTS

Default Arguments (การกำหนดค่าเริ่มต้น)

```
void myFunction(int x, int y = 0) {
  // ทำสิ่งที่ต้องการกับ x และ y
  cout << x + y << "\n";
}

int main() {
  // y จะใช้ค่าเริ่มต้น 0
  myFunction(5);
  // y จะใช้ค่าที่ถูกส่งเข้ามา 20
  myFunction(10, 20);
  return 0;
}
```

OUTPUT

```
5 // ในส่วนของ y ไม่มีการส่งค่าข้อมูลเข้ามาจึงให้ y เท่ากับ 0 แทน

30 // ในส่วนของ y มีการส่งค่ามา ให้ y มีค่าเท่ากับ 20 จึงเป็น 10 + 20
```

FUNCTION ARGUMENTS

Variadic Functions (ฟังก์ชันที่รองรับจำนวนอาร์กิวเมนต์ที่ไม่แน่นอน)

```
#include <cstdarg>
int sum(int count, ...) {
  int result = 0;
  va_list args;
  va_start(args, count);
  for (int i = 0; i < count; ++i) {
    result += va_arg(args, int);
  }
  va_end(args);
  return result;
}

int main() {
  cout << sum(4, 20, 20, 30, 40); // 60
  return 0;
}
```

OUTPUT

110 // ในส่วนของค่า parameter ตัวแรก จะเป็นค่าที่กำหนดว่าจะมีการวน loop กี่ครั้ง และค่า parameter ตัวที่ 2 - 5 จะเป็นค่าที่จะนำมาประมวลผล

FUNCTION ARGUMENTS

Function Overloading (การทำซ้ำฟังก์ชัน)

```
int add(int a, int b) {
  return a + b;
}

double add(double a, double b) {
  return a + b;
}

int main() {
  int sum_int = add(5, 10);
  double sum_double = add(5.5, 10.5);
  cout << sum_int << "\n";
  cout << sum_double << "\n";
  return 0;
}
```

OUTPUT

```
15 // เป็นค่าที่ได้รับจาก int add
16 // เป็นค่าที่ได้รับจาก double add
```

2

LIBRARY

C++ มีหลายไลบรารี *(Library)* ที่มีฟังก์ชันต่าง ๆ ที่สามารถใช้เพื่อช่วยในการพัฒนาโปรแกรมได้มากมาย นอกจากนี้ยังมีไลบรารีมาตรฐานที่รวมอยู่ใน *C++ Standard Library* ด้วย ซึ่งประกอบด้วยลักษณะต่าง ๆ เช่นคอลเลกชัน, การทำงานกับสตริง, การจัดการไฟล์, การทำงานกับเวลา, และอื่น ๆ นอกจากนี้ยังมีไลบรารีเสริม *(third-party libraries)* ที่สามารถใช้เพื่อขยายความสามารถของภาษา *C++* ได้

STL (STANDARD TEMPLATE LIBRARY)

ไลบรารีนี้มีคอลเลกชันของคลาสและฟังก์ชันที่มีอยู่ใน C++ Standard Library และมีโครงสร้างที่มีประสิทธิภาพ ได้แก่ vector, list, queue, stack, algorithm, และอื่น ๆ

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

int main() {
  vector<int> numbers = {5, 2, 8, 1, 7};

  sort(numbers.begin(), numbers.end());

  for (int num : numbers) {
    cout << num << " ";
  }

  return 0;
}
```

OUTPUT

```
1 2 5 7 8
```

IOSTREAM

ไลบรารีนี้ให้ฟังก์ชันสำหรับการทำงานข้อมูลที่สามารถนำเข้าและส่งออกได้, เช่น cout, cin, cerr, และ clog.

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

int main() {
  int number;

  cout << "Enter a number: ";
  cin >> number;

  cout << "You entered: " << number;

  return 0;
}
```

OUTPUT

```
5 // ถ้าหากป้อนเลข 5 จะแสดงผลลัพธ์เลข 5
```

CMATH

ไลบรารีที่ให้ฟังก์ชันทางคณิตศาสตร์, เช่นฟังก์ชันทางคณิตศาสตร์พื้นฐาน, ฟังก์ชันทางเลขศาสตร์, และอื่น ๆ

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

int main() {
  double x = 80.0;

  double result = std::sqrt(x);

  cout << "Square root of " << x << " is: " <<
result;

  return 0;
}
```

OUTPUT

```
Square root of 80 is: 8.94427
```

STRING

ไลบรารีที่ให้ฟังก์ชันทำงานกับข้อมูลประเภทสตริง, เช่นการเชื่อมต่อสตริง, การค้นหา, และการเปรียบเทียบ

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

int main() {
  string greeting = "Hello, ";
  string name = "John";
  string message = greeting + name;

  cout << message;

  return 0;
}
```

OUTPUT Hello, John

ALGORITHM

ไลบรารีนี้มีฟังก์ชันที่เกี่ยวข้องกับการประมวลผลข้อมูล, เช่นการเรียงลำดับ (sorting), การสลับ (swapping), และอื่น ๆ

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

int main() {
  vector<int> numbers = {5, 2, 8, 1, 7};

  swap(numbers[0], numbers[1]);

  for (int num : numbers) {
    cout << num << " ";
  }

  return 0;
}
```

OUTPUT

```
2 5 8 1 7
```

FSTREAM

ไลบรารีที่ให้ฟังก์ชันทำงานกับไฟล์, เช่นการเปิด, ปิด, อ่าน, และเขียน

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

int main() {

  ofstream outputFile("example.txt");
  outputFile << "Hello, C++ File I/O!";
  outputFile.close();

  ifstream inputFile("example.txt");
  string content;

  getline(inputFile, content);

  cout << content;

  return 0;
}
```

OUTPUT

Hello, C++ File I/O!

3

VARIABLE SCOPE

ในภาษา *C++*, ตัวแปรมีขอบเขต *(scope)* ซึ่งกำหนดถึงส่วนที่ตัวแปรสามารถเข้าถึงหรือใช้งานได้. *Scope* ของตัวแปรจะขึ้นอยู่กับที่ตัวแปรถูกประกาศ, และมีหลายระดับด้วยกัน. นี่คือกฎพื้นฐานเกี่ยวกับ *Scope* ใน *C++*:

GLOBAL SCOPE

ตัวแปรที่ถูกประกาศนอกฟังก์ชันหรือคลาสจะมี Global Scope ซึ่งหมายความว่าสามารถเข้าถึงได้ทั่วทุกที่ในโปรแกรม

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

int globalVar = 10;

int main() {
  cout << globalVar;
  return 0;
}
```

OUTPUT 10

LOCAL SCOPE

ตัวแปรที่ถูกประกาศภายในฟังก์ชันหรือบล็อกของโค้ดมี Local Scope, ซึ่งหมายความว่ามีอยู่เฉพาะในส่วนนั้น

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

int main() {
  int localVar = 5;
  cout << localVar;
  return 0;
}
```

OUTPUT 5

BLOCK SCOPE

ระดับของ Block Scope คือส่วนที่อยู่ในวงเล็บ {} ภายในฟังก์ชันหรือบล็อกของโค้ด. ตัวแปรที่ถูกประกาศภายในบล็อกจะมีขอบเขตเฉพาะภายในบล็อกนั้น

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

int main() {
  {
    int blockVar = 7;
    cout << blockVar;
  }
  return 0;
}
```

OUTPUT 7

FUNCTION PARAMETER SCOPE

พารามิเตอร์ที่ถูกประกาศในลำดับการเรียกฟังก์ชันมีขอบเขตเฉพาะในฟังก์ชันนั้น

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

void myFunction(int parameter) {
  cout << parameter;
}

int main() {
  int outsideVar = 42;
  myFunction(outsideVar);
  return 0;
}
```

OUTPUT 42

EXERCISE

1. จงเขียน function นำตัวเลขทั้งสองตัวมายกกำลัง
   Get(8, 2) output (8 ยกกำลัง 2 = 64)
   Get(9, 3) output (9 ยกกำลัง 3 = 729)
   Get(7, 5) output (7 ยกกำลัง 5 = 16,807)
   Get(6, 6) output (6 ยกกำลัง 6 = 279,936)

2. จงเขียน function ตรวจสอบว่าประโยคนั้นมีช่องว่างทั้งหมดเท่าไหร่
   Get(“Hello Thailand.”) output 1
   Get(“สัตว์ที่มี4 ขา ทุกตัวคือ แมว ปลามี4 ขา ปลาเป็นแมวหรือไม่”) output 8
   Get(“ประชากรบนโลก 1 ใน 3 เป็นลูกหลานของชาวมองโกล”) output 4

EXERCISE

3. จงเขียน function แทนที่ช่องว่างด้วยเครื่องหมาย “-”
   Get(“Hello Thailand.”) ผล “Hello-Thailand.”
   Get(“สัตว์ที่มี4 ขา ทุกตัวคือ แมว ปลามี4 ขา ปลาเป็นแมวหรือไม่”) output “สัตว์ที่มี-4-ขา-ทุกตัวคือ-แมว-ปลามี-4-ขา-ปลาเป็น แมวหรือไม่”
   Get(“ประชากรบนโลก 1 ใน 3 เป็นลูกหลานของชาวมองโกล”) output “ประชากรบนโลก-1-ใน-3-เป็นลูกหลานของชาวมองโกล”

4. จงเขียน function สลับตัวเลขเพื่อทำการเข้ารหัสลับของ
   Get(“01234”) output 56789
   Get(“98654”) output 43210
   Get(“06284”) output 51739

EXERCISE

5. จงเขียน function เมื่ออเมริกาต้องการส่งข้อความไปให้กับประเทศอินเดียแต่ไม่อยากให้ประเทศอื่นรู้ว่าส่งข้อความอะไรไป อเมริกาจึงต้องมีการเข้ารหัสลับโดยใช้ตัวอักษรที่เป็นตรงกันข้าม ภาษาอังกฤษมีทั้งหมด 26 ตัวอักษร A อยู่ต าแหน่งที่1 ซึ่งจะคู่กับ ตัวอักษรที่อยู่ต าแหน่งที่ 14 นั้นคือ N ถ้าหากอเมริกาส่งตัวอักษรตัว A แสดงอินเดียต้องรับรู้ว่าเป็น N
   Get(“A”) output N
   Get(“B”) output O
   Get(“URYYB VAQVN”) output HELLO INDIA

6. มีหอยทากอยู่1 ตัว อยู่ล่างบ่อน ้าจากก้นบ่อน ้า ถึง ปากบ่อน้ำมีระยะทางทั้งหมด 57 เมตร ตอนกลางวันหอยทากสามารถเดินขึ้นไปได้ 3 เมตร ตอนกลางคืนหอยทากจะลื่นลงมา 1.5 เมตร หอยทากต้องใช้เวลากี่วันถึงจะเดินกี่วันถึงจะขึ้นมาถึงปากบ่อน้ำได้

EXERCISE

7. โจเซฟผู้รอดชีวิตเข้าไปเข้าร่วมเกมส์ เกมส์หนึ่งและมีผู้เข้าร่วมคนอีก 15 คน นามสมมุติชื่อว่า A B C D E F G H I J K L O P Z ซึ่งโจเซฟนั้นได้ไปเข้าแถวหลังคนที่ชื่อว่า F โดยผู้คุมของเกมส์นั้น ได้ นับตั้งแต่ A – Z แล้วแน่นอนโจเซฟอยู่ในนั้นด้วย ถ้าหากจ านวนนับนั้นยังไม่ถึงตามที่ผู้คุมเกมส์ก าหนดนั้นจะวนกลับมาหัว แถวใหม่แล้วนับต่อไปเรื่อยๆ ครั้งนี้ผู้คุมนับไป 333 ครั้ง จะหาว่าใครจะถูกคัดออกในเกมส์นี

8. จากข้อที่ 7 ให้สุ่มตำแหน่งของผู้เข้าแข่งขัน
9. จากข้อ 7 และ 8 ให้เล่นเกมส์เหมือนเดิมแต่ให้คัดออกเรื่อยๆ จนเหลือคนสุดท้ายแล้วดูว่าใครจะเป็นผู้รอดชีวิตคนสุดท้าย

EXERCISE

10. มีอูฐ 31 ตัวเดินทางไปยังทิศเหนือขึ้นไป 180 กิโลเมตรเพื่อไปหาแหล่งน้ำการเดินทางวันที่เป็นเลขคี่นั้น เดินทางได้ 12 กิโลเมตรต่อวัน วันที่เป็นเลขคู่นั้น เดินทางได้11 กิโลเมตรต่อวัน แต่ๆ ทุกๆ 3 วันจะมีลูกอูฐตัวหนึ่งที่ป่วยท าให้ วันนั้นไม่สามารถเดินทางได้จะหาว่าอูฐทั้งหมดจะสามารถเดินไปถึงที่หมายได้ภายในกี่วัน

11. รับเวลาเป็น hh:mm จงหาว่าเข็มนาฬิกาเข็มสั้นกับเข็มยาวห่างกันกี่องศา 09:00 output 90 องศา 17:30 ผล 15 องศา
    Get(“09”, “00”) output 90 องศา
    Get(“17”, “30”) output 15 องศา

THANK YOU

HOPE THIS IS A GOOD START FOR YOU IN LEARNING PROGRAMMING.
GOOD LUCK ☘

- Function Argumentsin C++
- Library Functions in C++
- Variable Scope in C++
- Exercise with C++