



โปรแกรมภาษา

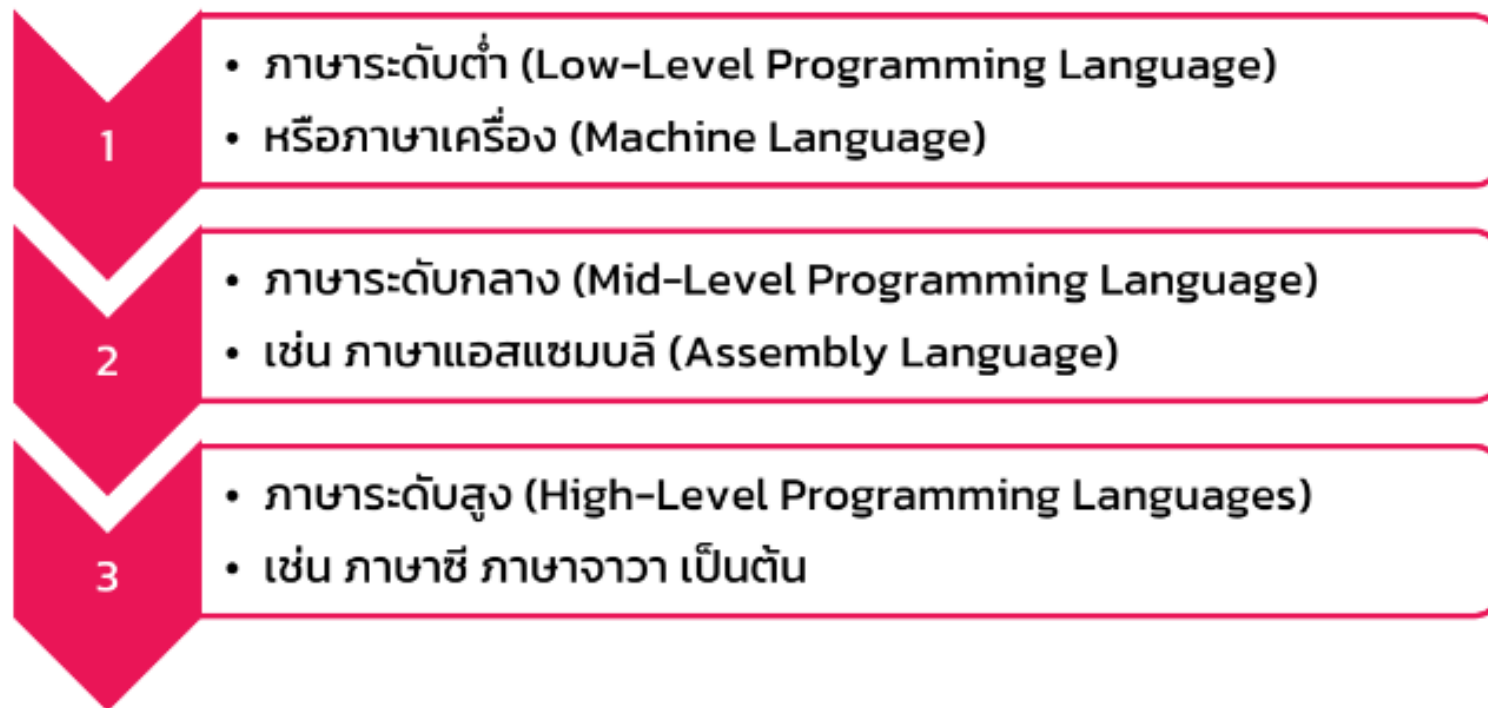
(Programming Languages)

ที่มา: เอกสารประกอบการสอนค่ายโอลิมปิกวิชาการ สอวน.

วิชาคอมพิวเตอร์ ค่าย 1 ศูนย์โรงเรียนสามเสนวิทยาลัย ปีการศึกษา 2564

1.โปรแกรมภาษา

ใช้ในการพัฒนาโปรแกรมประยุกต์สำหรับงานเฉพาะตามที่ต้องการ



ภาษาเครื่อง (Machine Language)

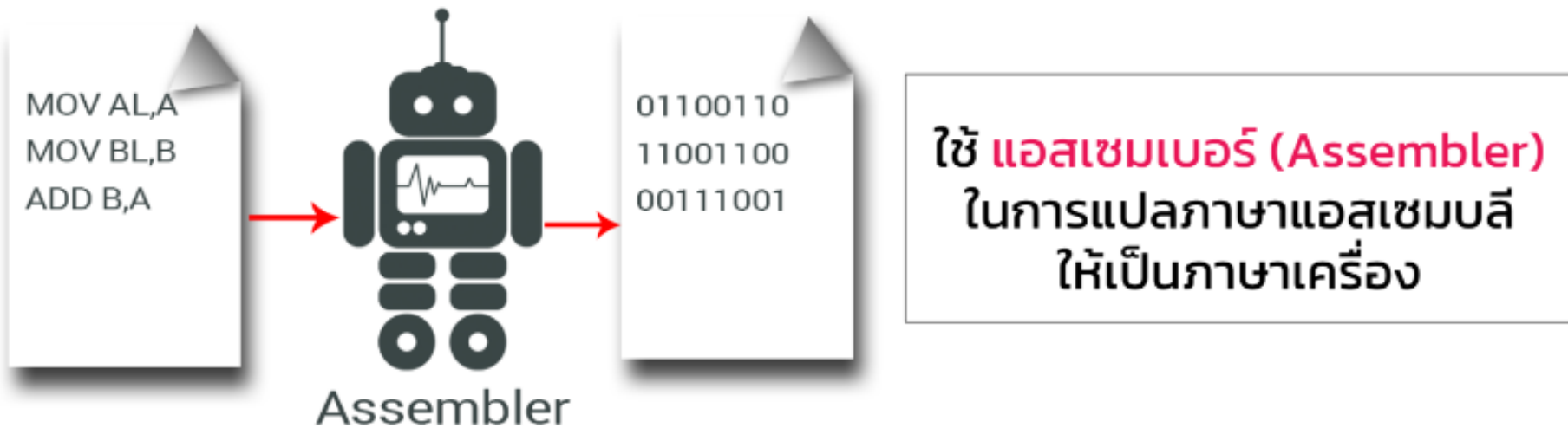
เป็นภาษาที่คอมพิวเตอร์เข้าใจ ซึ่งเขียนเป็นรหัสเลขฐาน 2 (0/1) และคำสั่งมีความเกี่ยวข้องกับอุปกรณ์ของคอมพิวเตอร์โดยตรง แต่มนุษย์เข้าใจภาษาเครื่องได้ยาก ดังนั้นการเขียนโปรแกรมด้วยภาษาเครื่องจึงยากมาก



ภาษาแอสเซมบลี (Assembly Language)

เป็นภาษาที่เขียนโดยใช้ภาษาอย่างง่ายที่มนุษย์เข้าใจ แทนการใช้รหัสเลขฐาน 2

- แต่ออกแบบมาเฉพาะสำหรับคอมพิวเตอร์แต่ละแบบ
- และผู้เขียนโปรแกรมยังต้องทราบข้อมูลที่เกี่ยวข้องกับอุปกรณ์ของคอมพิวเตอร์



ภาษาระดับสูง (High-Level Languages)

- ใช้ภาษาที่มนุษย์เข้าใจ (English-like language) เช่น **C C++ JAVA Python**
- ใช้ คอมไพเลอร์ (Compiler) หรือ Interpreter ในการแปลภาษาระดับสูงให้เป็นภาษาเครื่อง



- **Compiler** แปลทั้งโปรแกรม (เช่น Pascal, C, ...)
- **Interpreter** แปลทีละบรรทัด (เช่น Basic, ...)

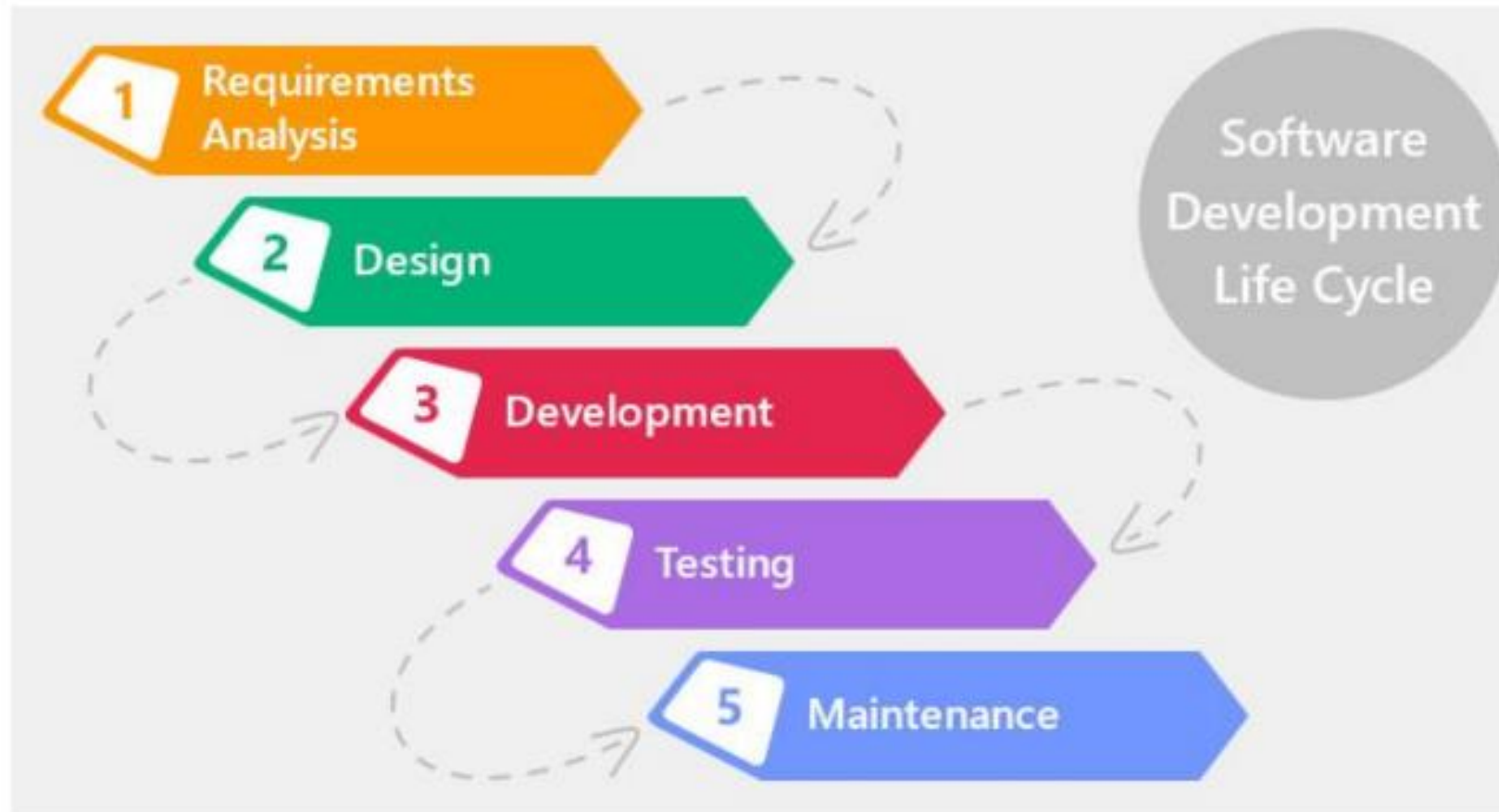
Compiler

1010
0111

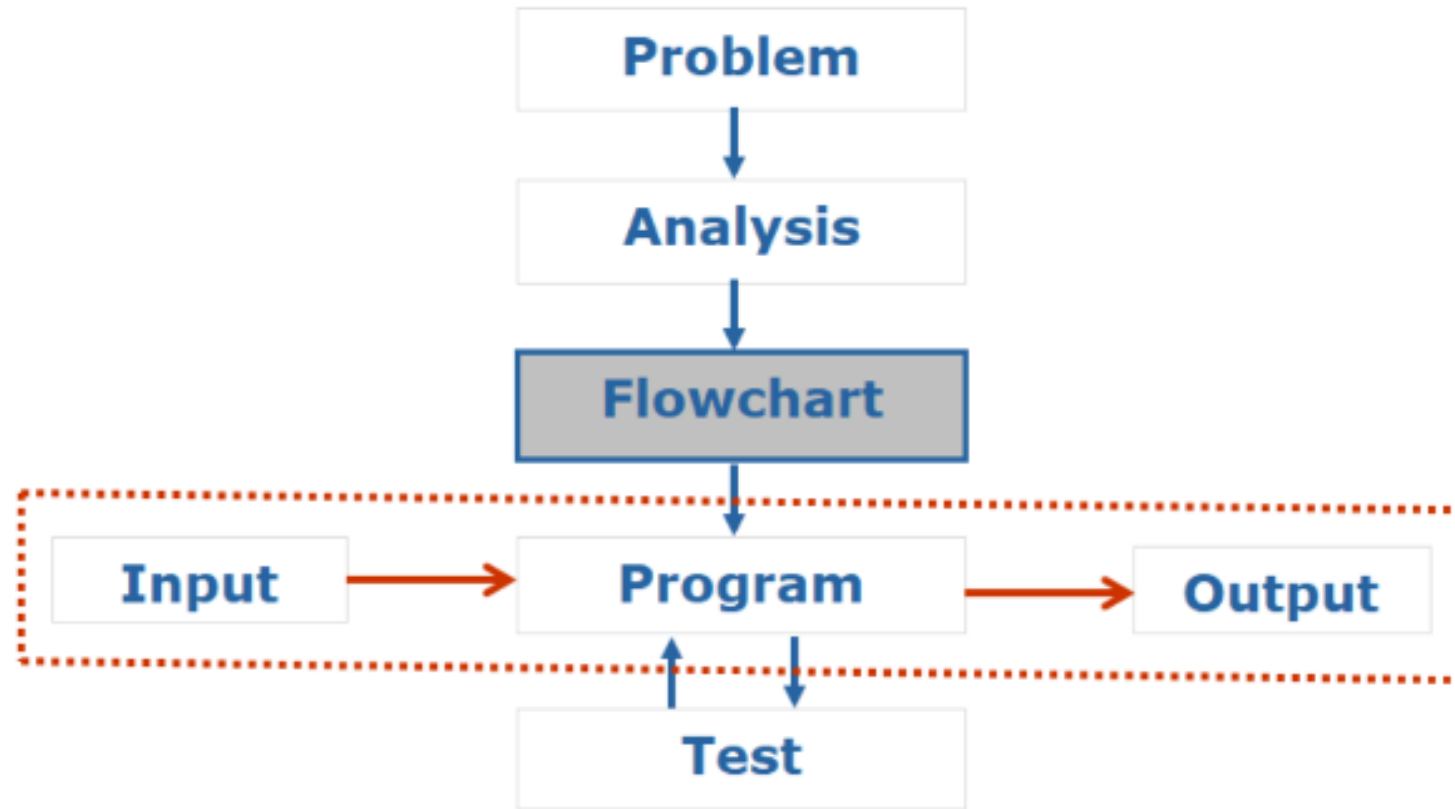
02

การพัฒนาโปรแกรม

ขั้นตอนการพัฒนาโปรแกรม



สรุป 5 ขั้นตอนการพัฒนาโปรแกรม



2.1กำหนด-วิเคราะห์ปัญหา (Analysis)

กำหนดขอบเขตของปัญหา

จะให้คอมพิวเตอร์ทำอะไร (What?)

วิเคราะห์ปัญหา (Problem analysis)

input : ลักษณะข้อมูลนำเข้า

process : วิธีการประมวลผล (how?)

output : ลักษณะของผลลัพธ์ที่ต้องการ

2.1 กำหนด-วิเคราะห์ปัญหา (Analysis)

ตัวอย่าง

การกำหนด-วิเคราะห์ปัญหาในชีวิตประจำวัน

อยากกินกระเพราไก่ไข่ดาว



2.1 กำหนด-วิเคราะห์ปัญหา (Analysis)

ตัวอย่าง

การกำหนด-วิเคราะห์ปัญหาในชีวิตประจำวัน

กำหนดขอบเขตของปัญหา

ต้องการกระเปาะไม้ไผ่ 1 จาน

วิเคราะห์ปัญหา (Problem analysis)

input : ใบกระเปาะ เนื้อไม้ เครื่องปรง ไม้ไผ่

process : นำวัสดุมาตัดพร้อมปรงรสในกระทะ ทอดไม้ไผ่

output : กระเปาะไม้ไผ่

2.1 กำหนด-วิเคราะห์ปัญหา (Analysis)

ตัวอย่างการกำหนด-วิเคราะห์ปัญหา

ออกแบบโปรแกรมให้คอมพิวเตอร์ทำงานเป็น
เครื่องคิดเลขอย่างง่าย โดยรับข้อมูลตัวเลข 2 จำนวน (X, Y)
และแสดงผลบวกทางจอภาพ

กำหนดขอบเขตของปัญหา

คำนวณผลบวกของ 2 จำนวน

วิเคราะห์ปัญหา (Problem analysis)

input : รับข้อมูลตัวเลข (X, Y) จากคีย์บอร์ด

process : คำนวณ $sum = X + Y$

output : แสดงผลบวก (sum) ทางจอภาพ

2.1 กำหนด-วิเคราะห์ปัญหา (Analysis)

ตัวอย่างการกำหนด-วิเคราะห์ปัญหา

ออกแบบโปรแกรมให้คอมพิวเตอร์รับข้อมูลตัวเลข 3 จำนวน คำนวณค่าเฉลี่ย และแสดงค่าเฉลี่ย ทางจอภาพ

กำหนดขอบเขตของปัญหา

คำนวณค่าเฉลี่ยของ 3 จำนวน $((X1+X2+X3)/3)$

วิเคราะห์ปัญหา (Problem analysis)

input : รับข้อมูลตัวเลข (X1, X2, X3)

process : คำนวณ $sum = X1 + X2 + X3$
 $mean = sum/3$

output : แสดงค่าเฉลี่ย (mean) ทางจอภาพ

2.2 การออกแบบขั้นตอนการแก้ปัญหา (Design)

คือ การนำปัญหาที่วิเคราะห์ได้จากขั้นตอนที่ 1 มาวางแผน
อย่างเป็นขั้นตอนว่าจะต้องเขียนโปรแกรมเพื่อแก้ปัญหายังไง การ
วางแผนอย่างเป็นขั้นตอนนี้ เรียกว่า **อัลกอริทึม (Algorithm)** ซึ่ง
อัลกอริทึมแบ่งออกเป็น 2 รูปแบบ คือ

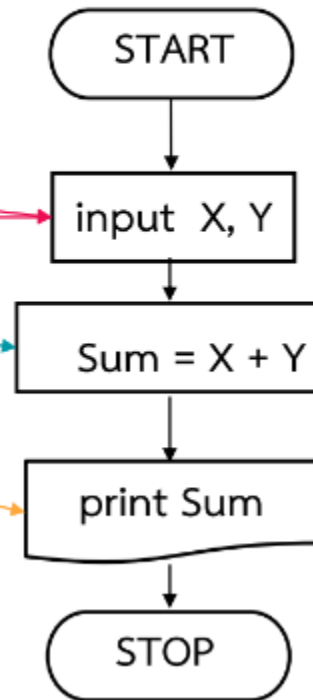
- **ซูโดโค้ด (Pseudo code)**
- **ผังงาน (Flowchart)**

2.2 การออกแบบขั้นตอนการแก้ปัญหา (Design)

ซูโดโค้ด (Pseudo code)




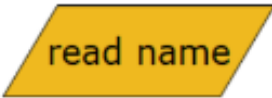

```
START  
READ X  
READ Y  
COMPUTE SUM = X+Y  
PRINT SUM  
STOP
```

ผังงาน (Flowchart)




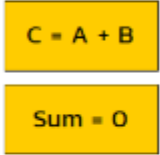


2.2 การออกแบบขั้นตอนการแก้ปัญหา (Design)

สัญลักษณ์ในผังงาน

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	คำอธิบาย
	การเริ่มต้นหรือสิ้นสุดการเขียนผังงาน (Terminal)	 	1. เริ่มต้นผังงาน 2. จบผังงาน
	รับข้อมูลหรือแสดงข้อมูลโดยไม่ระบุชื่อ (Input/output)	 	1. รับค่าใส่ในตัวแปรชื่อ name 2. แสดงค่าจากตัวแปร area




2.2 การออกแบบขั้นตอนการแก้ปัญหา (Design)

สัญลักษณ์ในผังงาน

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	คำอธิบาย
	การประมวลผล (Process)	 <pre>graph TD; A["C = A + B"] --> B["Sum = 0"];</pre>	1. คำนวณ $A + B$ และเก็บไว้ใน C 2. กำหนดค่า sum เท่ากับ 0
	การเปรียบเทียบ หรือตัดสินใจ (Compare / Decision)	 <pre>graph TD; Entry(()) --> D{"i <= 10"}; D -- true --> P[/แสดง i/]; P --> Exit(()); D -- false --> Exit;</pre>	เปรียบเทียบถ้า i มีค่า น้อยกว่าหรือเท่ากับ 10 - เป็นจริง พิมพ์ค่า i เสร็จแล้วไปทำ คำสั่งอื่น ๆ - เป็นเท็จ ไปทำคำสั่งอื่น ๆ




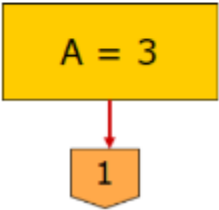
2.2 การออกแบบขั้นตอนการแก้ปัญหา (Design)

สัญลักษณ์ในผังงาน

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	คำอธิบาย
	การแสดงผลลัพท์ทาง เครื่องพิมพ์ (Document)		พิมพ์ค่า A ทางเครื่องพิมพ์
	การแสดงผลลัพท์ทาง จอภาพ (Display)		แสดงค่า A, B บนจอภาพ

2.2 การออกแบบขั้นตอนการแก้ปัญหา (Design)

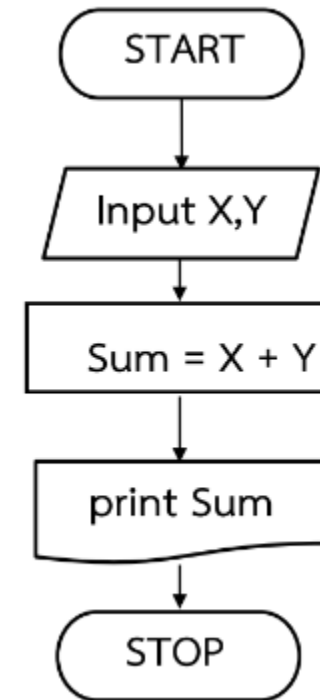
สัญลักษณ์ในผังงาน

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	คำอธิบาย
	จุดต่อเนืองในหน้าเดียวกัน (In-Page connector)		หลังจากพิมพ์ค่า A แล้วให้ทำตามจุดต่อเนือง A ซึ่งอยู่ในหน้าเดียวกัน
	จุดต่อเนืองที่อยู่คนละหน้า (Off-Page Connector)		หลังจากกำหนดค่า A เท่ากับ 3 ให้ทำตามจุดต่อเนืองชื่อ 1 ซึ่งไม่ได้อยู่ในหน้าเดียวกัน

2.2 การออกแบบขั้นตอนการแก้ปัญหา (Design)

หลักการจัดภาพผังงาน

ทิศทางของผังงานจะเริ่มจากส่วนบนของหน้ากระดาษลงมายังส่วนล่าง และจากซ้ายมือไปของหน้ากระดาษ และควรเขียนเครื่องหมายลูกศรกำกับทิศทางไว้ด้วย

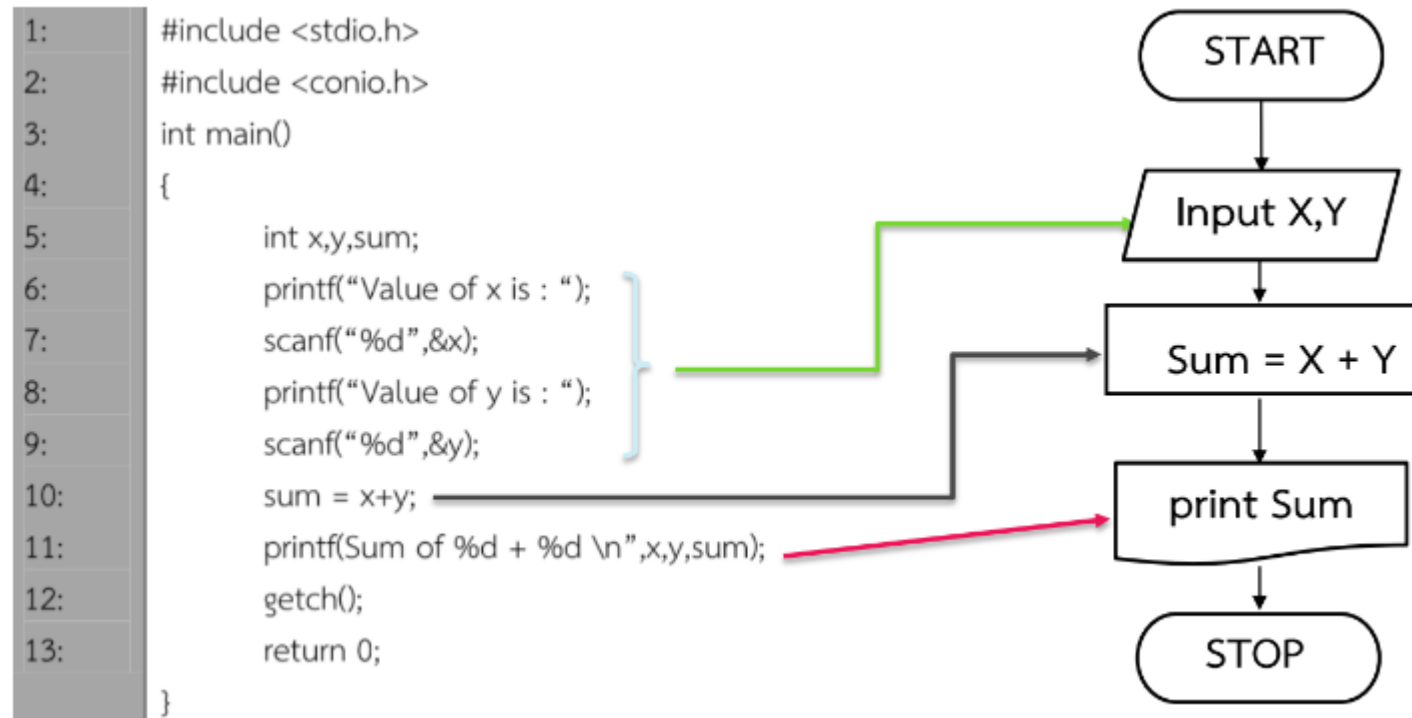


2.3 การพัฒนาโปรแกรม (Development)

เป็นการนำอัลกอริทึมจากขั้นตอนที่ 2 มาเขียนโปรแกรมให้ถูกต้องตามหลักไวยากรณ์ (Syntax) ของภาษาซี จากโจทย์สามารถเขียนโปรแกรมได้ดังนี้

```
1: #include <stdio.h>
2: #include <conio.h>
3: int main()
4: {
5:     int x,y,sum;
6:     printf("Value of x is : ");
7:     scanf("%d",&x);
8:     printf("Value of y is : ");
9:     scanf("%d",&y);
10:    sum = x+y;
11:    printf("Sum of %d + %d \n",x,y,sum);
12:    getch();
13:    return 0;
}
```

2.3 การพัฒนาโปรแกรม (Development)



2.4 การทดสอบโปรแกรม (Testing)

เป็นการประกันคุณภาพของโปรแกรม ค้นหาข้อผิดพลาด ป้องกันการเกิดข้อผิดพลาดของซอฟต์แวร์ รวมไปถึงตรวจสอบว่าโปรแกรมที่พัฒนานั้นเป็นไปตาม Requirement หรือไม่ ในการทดสอบนั้นอาจมีการสร้าง Test case หรือเป็นกรณีที่ผู้ใช้งานจะใช้งานระบบและ Expected results คือผลที่คาดหวังจากการทดสอบ



2.5 การบำรุงรักษาซอฟต์แวร์ (Maintenance)

เป็นขั้นตอนการบำรุงรักษาระบบ
ต่อเนื่องหลังจากเริ่มดำเนินการ ผู้ใช้งาน
อาจจะพบกับปัญหาที่เกิดขึ้นภายหลัง เช่น
ต้องการ Feature เพิ่มเติม ซอฟต์แวร์ทำงาน
ไม่เป็นไปตามความคาดหวัง หรือซอฟต์แวร์มี
Bug อยู่

เมื่อเกิดปัญหาเหล่านี้เกิดขึ้นก็ต้องมี
การแก้ไข จึงควรกำหนดแผนการแก้ปัญหา
และติดตามผล เพื่อทำการปรับปรุงให้ผู้ใช้งาน
เกิดความพึงพอใจ



3. การโปรแกรมเบื้องต้น

- การดาวน์โหลดและติดตั้งโปรแกรม
- เริ่มต้นเขียนภาษาซี
- คำสั่งแสดงผลทางจอภาพ
- ประเภทของข้อมูลและตัวแปร
- ตัวดำเนินการ
- คำสั่งรับข้อมูล

การโปรแกรมเบื้องต้น

การ Download และติดตั้งโปรแกรม IDE

- โปรแกรม Dev-C++

https://drive.google.com/file/d/1_d80TQMZLFksjBABQeSstyObEduHiVuo/view

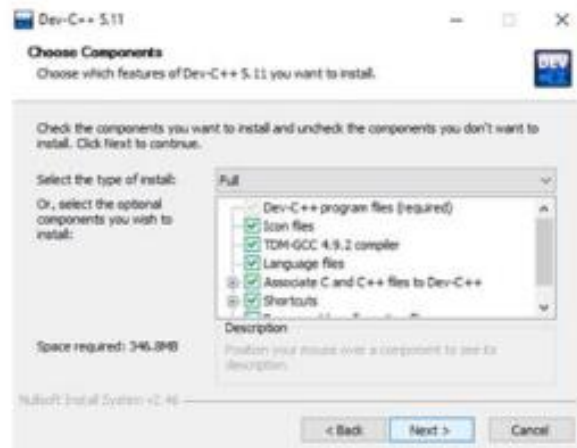
หรือเขียนออนไลน์บนเว็บ <https://www.onlinegdb.com/>

3.1 การติดตั้งโปรแกรม IDE โปรแกรม Dev-C++

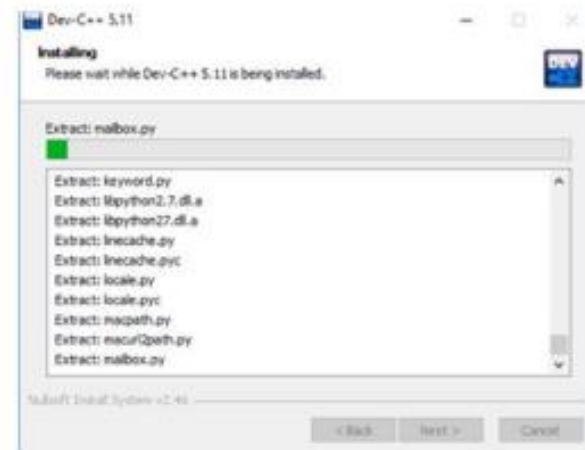
• 1) ดับเบิลคลิก



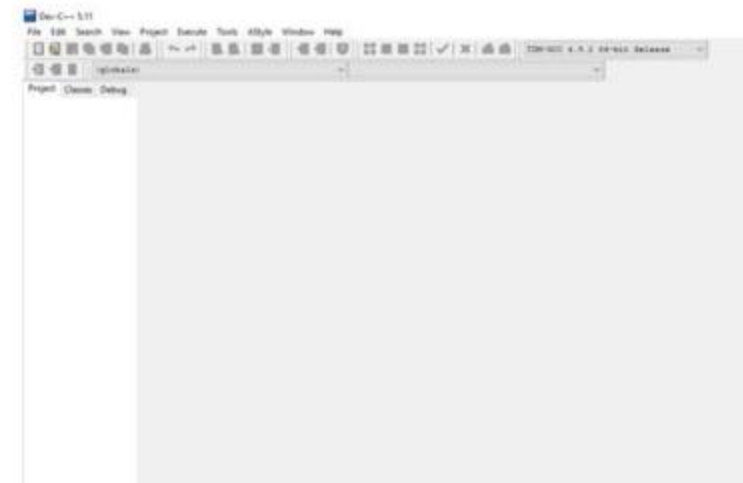
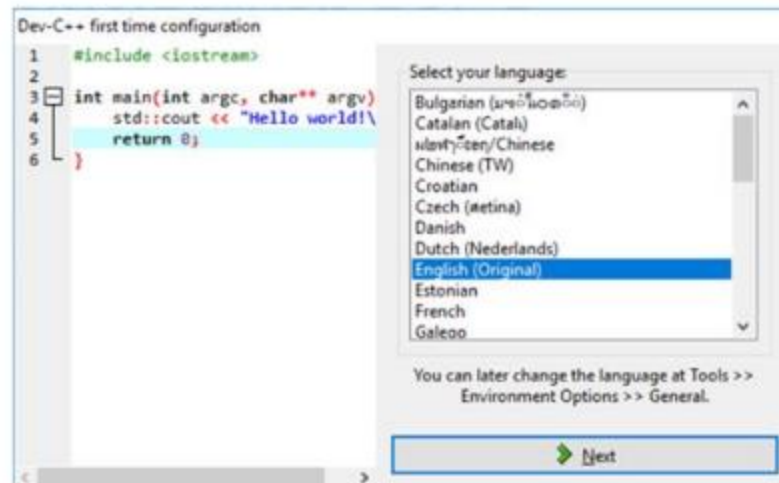
2) คลิก Next



3) Install

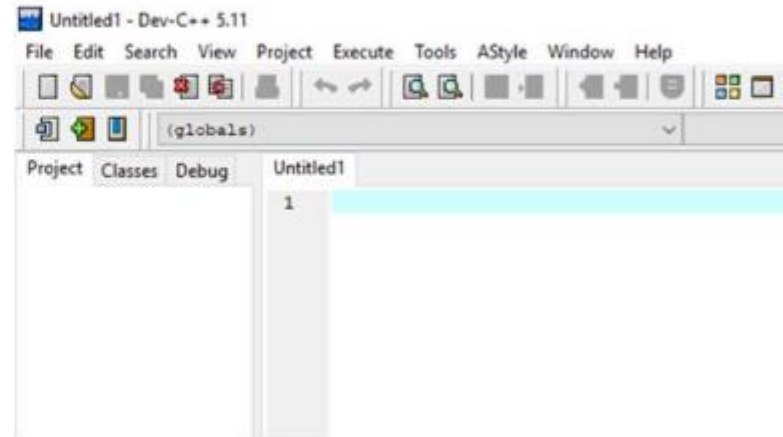
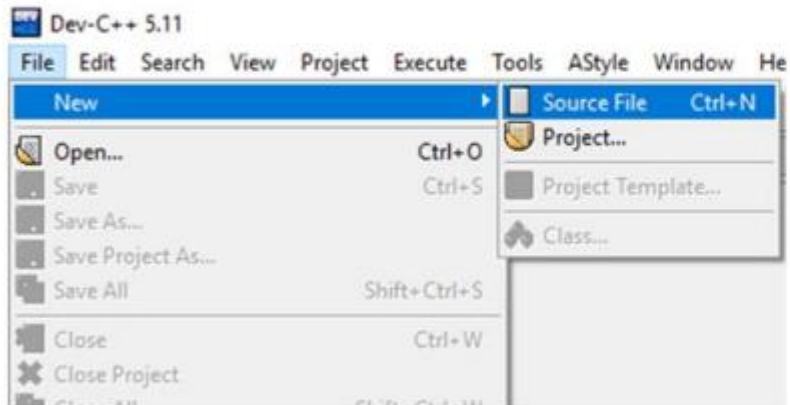


3.1 การติดตั้งโปรแกรม IDE โปรแกรม Dev-C++



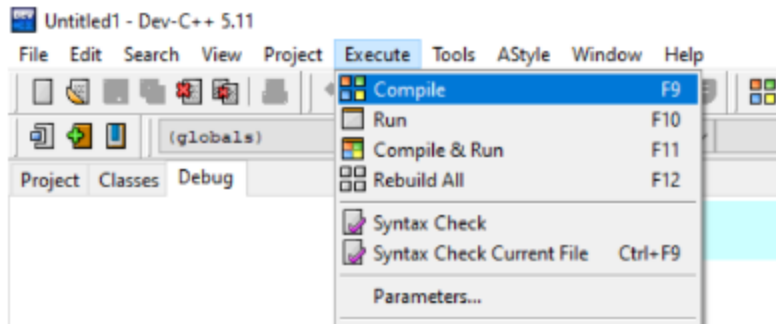
3.1 การติดตั้งโปรแกรม IDE โปรแกรม Dev-C++

เลือก File > Source File

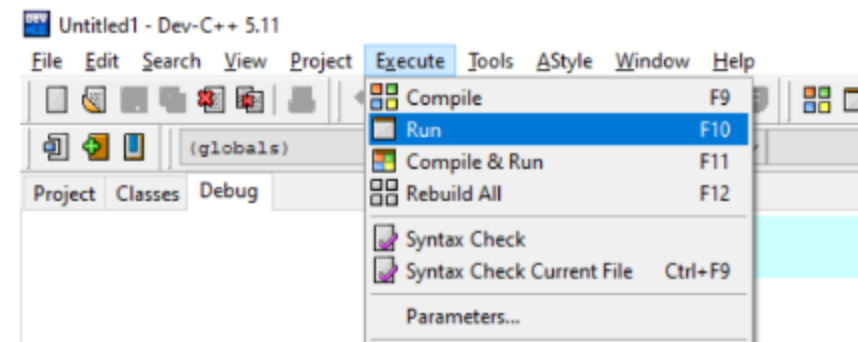


3.1 การติดตั้งโปรแกรม IDE โปรแกรม Dev-C++

1) Compile

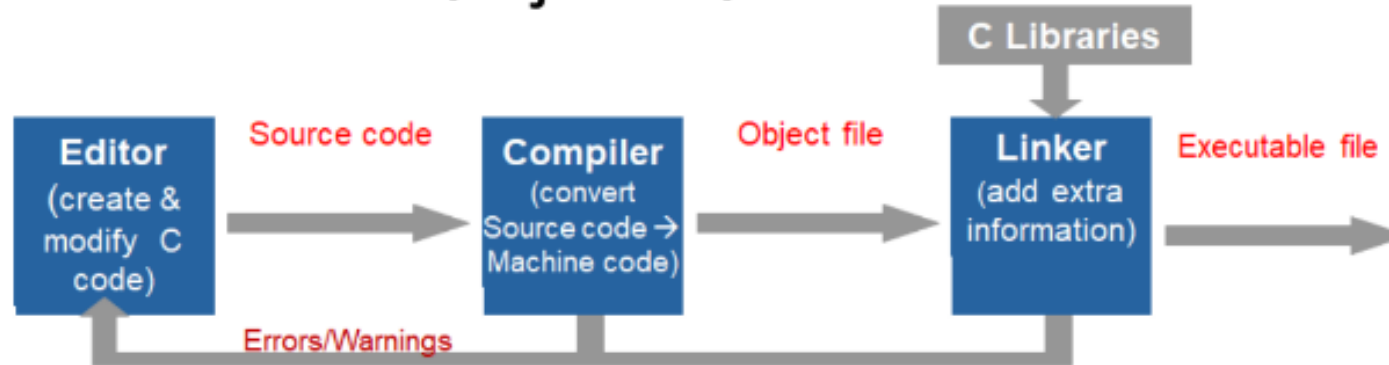


2) Run



3.1 การติดตั้งโปรแกรม IDE โปรแกรม Dev-C++

ขั้นตอนการแปล Source code (file) ของโปรแกรมภาษา C ให้เป็น Machine code (Object file)



โปรแกรมภาษา C จะอยู่ในรูปแบบของฟังก์ชัน ซึ่งมีอย่างน้อยหนึ่งฟังก์ชัน (คือ **main**)

3.2 เริ่มต้นการเขียนโปรแกรม

โครงสร้างโปรแกรม

Preprocessor directive

global definitions

main function {

local definitions

statement

}

Head

body

```
#include <stdio.h>
void main()
{
    printf("Computer");
    return 0;
}
```

Computer

Process exited after 1.81 seconds with r
Press any key to continue . . .

62

3.2 เริ่มต้นการเขียนโปรแกรม

โครงสร้างโปรแกรม

```
#include <stdio.h>
int main() {
    printf("Computer");
    return 0;
}
```

คำสั่ง include ให้นำไฟล์ stdio.h มาร่วม
ด้วย เพื่อเรียกใช้ฟังก์ชันต่าง ๆ
หรืออาจเรียกไฟล์ .h อื่น ๆ อีกก็ได้
โดยไฟล์ที่เรียกจะต้องอยู่ใน " " หรือ < >

ฟังก์ชันในภาษาซี จบด้วยเครื่องหมาย ;
ในที่นี้จะให้แสดงข้อความทางจอภาพ
และขึ้นบรรทัดใหม่

void ที่อยู่หน้า main จะบอกว่าเมื่อทำตั้งแต่
{ ถึง }
ไม่ต้องคืนค่า หรือไม่ต้องทำอะไรต่อไปอีก

3.2 เริ่มต้นการเขียนโปรแกรม

ข้อควรระวังการ save ไฟล์

```
#include <stdio.h>
int main() {
    printf("Computer");
    return 0;
}
```

fileName.c

```
#include <stdio.h>
int main() {
    printf("Computer");
    return 0;
}
```

fileName.cpp

3.2 เริ่มต้นการเขียนโปรแกรม

C library

`include <Library>`

`include "Library"`

ไฟล์นามสกุล .h เป็น Header file หรือไฟล์ส่วนหัวที่รวบรวมคำสั่งต่าง ๆ ของภาษาซีเอาไว้ เพื่อให้ผู้เขียนสามารถใช้ฟังก์ชันได้ มีอยู่หลายไฟล์เช่น

- **stdio.h** เก็บฟังก์ชันที่ใช้งานทั่วไป เช่น printf , scanf โปรแกรมส่วนมาก จะใช้ไฟล์นี้
- **conio.h** เก็บฟังก์ชันควบคุมการแสดงผลต่าง ๆ
- **string.h** เก็บฟังก์ชันที่ใช้ในการประมวลผลกับข้อความ
- **math.h** เก็บฟังก์ชันที่ใช้ในการคำนวณทางคณิตศาสตร์ เช่น sin,cos,log

Libery อื่น ๆ เพิ่มเติม

C library - C++ Reference
(cplusplus.com)

3.3 คำสั่งแสดงผล

คำสั่ง **printf** ต้อง `include<stdio.h>`

รูปแบบคำสั่ง

```
printf("control string", variable,...) ;
```

Variable

เป็นตัวแปรใช้เก็บค่า (ที่เปลี่ยนแปลงได้) ใน memory ในขณะประมวลผล

ตัวอย่างการใช้งาน

control string ประกอบด้วย

- ข้อความอธิบาย เช่น `printf("C Programming");`
- **%format** เช่น `printf("SUM = %d\n", sum);`
- **Escape sequence** เช่น `\n` (new line)

3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งานคำสั่ง printf

```
1 #include <stdio.h>
2 int main()
3 {
4     printf ("Computer");
5     printf("Programming");
6     return 0;
7 }
```

ผลการรัน :

ComputerProgramming

3.3 คำสั่งแสดงผล

รหัสควบคุมการแสดงผล (Escape Sequence)

- `\n` ขึ้นบรรทัดใหม่
- `\t` เว้นช่องว่างเป็นระยะ 1 tab หรือหกตัวอักษร
- `\r` ให้เคอร์เซอร์ขึ้นบรรทัดใหม่
- `\f` เว้นช่องว่างเป็นระยะหนึ่งหน้าจอ
- `\"` พิมพ์ตัวอักษร " ในส่วน Control String
- `\\` พิมพ์ตัวอักษร \ ในส่วน Control String
- `%%` พิมพ์ตัวอักษร % ในส่วน Control String

3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งาน Escape Sequence

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("Computer\n");
5     printf("Programming\n");
6     return 0;
7 }
```

ผลการรัน :

```
Computer
Programming
```

3.3 คำสั่งแสดงผล

คอมเมนต์ (Comment)

คือส่วนที่เป็นหมายเหตุของโปรแกรม มีไว้เพื่อให้ผู้เขียนโปรแกรมใส่ข้อความอธิบายกำกับลงไปใน source code ซึ่งคอมไพเลอร์จะข้ามการแปลผลในส่วนที่เป็นคอมเมนต์นี้ คอมเมนต์ในภาษาซีมี 2 แบบคือ

- คอมเมนต์แบบ**บรรทัดเดียว** ใช้เครื่องหมาย //
- คอมเมนต์แบบ**หลายบรรทัด** ใช้เครื่องหมาย /* และ */

3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งาน comment

```
1 #include <stdio.h>
2 int main()
3 {
4     //printf ("Computer\n");
5     printf("Programming\n");
6     return 0;
7 }
```

ผลการรัน :

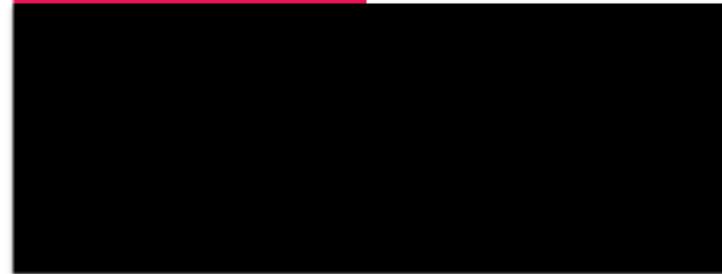
Programming

3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งาน comment

```
1 #include <stdio.h>
2 int main()
3 {
4     //printf ("Computer\n");
5     //printf("Programming\n");
6     return 0;
7 }
```

ผลการรัน :



3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งาน comment

```
1 #include <stdio.h>
2 int main()
3 {
4     /*printf ("Computer\n");
5     printf("Programming\n");*/
6     return 0;
7 }
```

ผลการรัน :



3.3 คำสั่งแสดงผล

format code

%c

ใช้แสดงอักขระตัวเดียว (Char)

%s

ใช้แสดงชุดตัวอักษรหรือข้อความ (String)

%d

ใช้แสดงจำนวนเต็ม (Integer)

%f

ใช้แสดงเลขทศนิยม (Float)

%h

ใช้แสดง short interger

%u

ใช้แสดง unsigned int

%o

ใช้แสดงจำนวนเต็มเลขฐาน 8

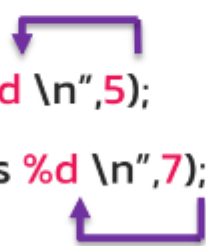
%x

ใช้แสดงข้อมูลเลขฐาน 16

3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งาน format code

```
1 #include <stdio.h>
2 int main()
3 {
4     printf ("First value is %d \n",5);
5     printf ("Second value is %d \n",7);
6     return 0;
7 }
```

A purple arrow starts from the first printf statement on line 4, goes down to the second printf statement on line 5, and then loops back up to the second printf statement, indicating the sequence of execution.

ผลการรัน :

```
First value is 5
Second value is 7
```

3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งาน format code

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("%s Smith %d %f %c", "Sam", 14, 51.5, 'M');
5     return 0;
6 }
```

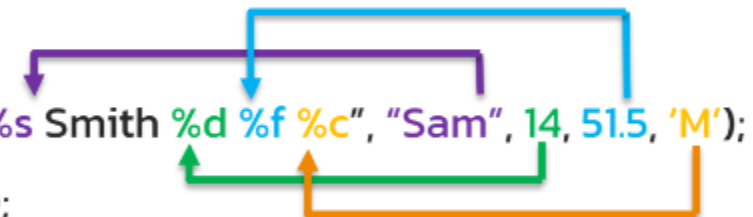
ผลการรับ :

Sam Smith 14 51.500000 X

3.3 คำสั่งแสดงผล

ตัวอย่างการใช้งาน format code

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("%s Smith %d %f %c", "Sam", 14, 51.5, 'M');
5     return 0;
}
```



ผลการรัน :

Sam Smith 14 51.500000 X

3.3 คำสั่งแสดงผล

ตัวอย่างการกำหนดจำนวนเลขทศนิยม

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("%s Smith %d %.2f %c", "Sam", 14, 51.5, 'M');
5     return 0;
6 }
```

ผลการรัน :

Sam Smith 14 51.50 M

3.4 ประเภทของข้อมูลและตัวแปร

ตัวแปร

ตัวแปร เป็นชื่อของหน่วยความจำที่ตำแหน่งต่าง ๆ ที่ผู้เขียนโปรแกรมกำหนด มีไว้สำหรับ**เก็บข้อมูลต่าง ๆ** ระหว่างการทำโปรแกรม

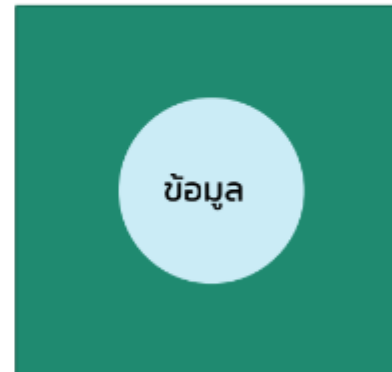


กล่อง : ตัวแปร

3.4 ประเภทของข้อมูลและตัวแปร

ตัวแปร

ตัวแปร เป็นชื่อของหน่วยความจำที่ตำแหน่งต่าง ๆ ที่ผู้เขียนโปรแกรมกำหนด มีไว้สำหรับ**เก็บข้อมูลต่าง ๆ** ระหว่างการทำโปรแกรม



กล่อง : ตัวแปร

3.4 ประเภทของข้อมูลและตัวแปร

ชนิดข้อมูล

ตัวอย่างของข้อมูล

“Hello”



ข้อความ



25



ตัวเลข



3.45



ทศนิยม



3.4 ประเภทของข้อมูลและตัวแปร

ชนิดข้อมูล

ชนิดข้อมูล	การประกาศ	ขนาด (byte)	ช่วงข้อมูล
ตัวอักษร	char	1	-128 ถึง 127
เลขจำนวนเต็ม	int	2	-32,768 ถึง 32,767
unsigned int	unsigned int	2	0 ถึง 65,535
long integer	long	4	สองพันล้าน
เลขทศนิยม	float	4	1.2×10^{-38} ถึง 3.4×10^{38}

ถ้าหากคอมพิวเตอร์ต่างกันขนาดก็อาจต่างกันไปด้วย

3.4 ประเภทของข้อมูลและตัวแปร

การประกาศตัวแปร

หลักการ

- ต้อง**ประกาศชนิดของ**ตัวแปรก่อนเรียกใช้
- ชนิดของตัวแปร ต้องสอดคล้องกับข้อมูลที่จะใช้เก็บ
เช่น เก็บเกรดเฉลี่ย ซึ่งเป็นทศนิยม ตัวแปรต้องเป็น
ชนิด float

3.4 ประเภทของข้อมูลและตัวแปร

กฎการตั้งชื่อตัวแปร

ตัวแปรจะเกิดจากเราเป็นคนกำหนดชื่อขึ้นมา

1. ชื่อจะประกอบขึ้นจาก ตัวอักษร ตัวเลข และ เครื่องหมายขีดเส้นใต้ เท่านั้น
เช่น Salary1 และ Salary_1
2. อักขระแรกของชื่อจะต้องเป็น ตัวอักษร หรือ เครื่องหมายขีดเส้นใต้ เท่านั้น
เช่น ASalary และ _Salary
3. ตัว พิมพ์ใหญ่ และ ตัวพิมพ์เล็ก ถือเป็นอักขระคนละตัวกัน
เช่น Salary และ SALARY
4. ชื่อจะต้อง ไม่ซ้ำ กับคำสงวน

3.4 ประเภทของข้อมูลและตัวแปร

กฎการตั้งชื่อตัวแปร

การตั้งชื่อตัวแปรต่อไปนี้ถูกหรือผิด

computer_room



_good



n-sync



User name



cat108



if



Year#



108cat



3.4 ประเภทของข้อมูลและตัวแปร

การประกาศตัวแปร

ตัวอย่างการประกาศตัวแปรตัวเดียว

```
#include<stdio.h>
int main(){
    int var;
    var=3;
    printf("%d",var);
    return 0;
}
```

รูปแบบการประกาศตัวแปร

ชนิดข้อมูล ชื่อตัวแปร;

จากตัวอย่าง

int var ;

ตัวแปร var เป็นชนิด int สำหรับเก็บจำนวนเต็ม

var

← ใช้เก็บข้อมูลชนิด int

หน่วยความจำ

3.4 ประเภทของข้อมูลและตัวแปร

การประกาศตัวแปร

```
#include<stdio.h>
int main(){
    int var1,var2;
    var1=3;
    var2=5;
    printf("%d\n",var1);
    printf("%d",var2);

    return 0;
}
```

รูปแบบการประกาศตัวแปร

ชนิดข้อมูล ชื่อตัวแปร,ชื่อตัวแปร;

จากตัวอย่าง

int var1,var2 ;

ตัวแปร var1 และ var2 เป็นชนิด int สำหรับเก็บจำนวนเต็ม

var1

← ใช้เก็บข้อมูลชนิด int

var2

← ใช้เก็บข้อมูลชนิด int

หน่วยความจำ

3.4 ประเภทของข้อมูลและตัวแปร

การประกาศตัวแปรและการกำหนดค่าเริ่มต้น

การประกาศตัวแปรและกำหนดค่าเริ่มต้น

รูปแบบ
ชนิด ตัวแปร = ค่าที่ต้องการ; Assignment Operator

- กำหนดค่าพร้อมกับประกาศตัวแปร หรือประกาศแล้วจึงกำหนดก็ได้ ตัวอย่างเช่น

```
int score;  
score = 20;
```

```
int score;  
score = 10+5;
```

```
int score=20;
```

```
int score=10+5;
```

3.4 ประเภทของข้อมูลและตัวแปร

การประกาศตัวแปร
และการกำหนดค่าเริ่มต้น

การกำหนดค่าให้ตัวแปร

รูปแบบ

ตัวแปร ค่าที่ต้องการกำหนด

```
var1 = 20
```

```
var2 = var1
```

รูปแบบ

ตัวแปร = ตัวแปร = ตัวแปร = ค่าที่ต้องการกำหนด

```
int var1 , var2 , var3
```

```
var1 = var2 = var3 = 20
```

3.4 ประเภทของข้อมูลและตัวแปร

ตัวแปรชนิดอักขระ (char)

```
char a, b;
```

```
a = 'T';
```

```
b = 'A';
```

```
c = 65;
```

กำหนดค่าตัวอักขระให้กับตัวแปรจะต้องอยู่ในเครื่องหมาย ''

ชวนคิด

```
int b;
```

```
b = 'a' + 3;
```

/* ค่าในตัวแปร b จะเท่ากับรหัส ASCII ของ a บวกกับ 3 */

3.4 ประเภทของข้อมูลและตัวแปร

ตัวแปรชนิดข้อความ (string)

ข้อความในภาษา C คือตัวอักษรหลาย ๆ ตัวมาต่อเรียงกัน เรียกว่าสตริง (string) เมื่อมีการประกาศ ตัวแปรประเภทนี้ คอมไพเลอร์จะนำตัวอักษรแต่ละตัวมาเก็บในหน่วยความจำแบบ char เรียงกันไปทีละตัว

รูปแบบคำสั่ง

```
char name[n] = "str"
```

name : ชื่อตัวแปร

n : ขนาดของข้อความ หรือจำนวนอักขระในข้อความ

str : ข้อความเริ่มต้นที่กำหนดให้กับตัวแปร โดยเขียนไว้ในเครื่องหมาย " "

3.4 ประเภทของข้อมูลและตัวแปร

ตัวแปรชนิดข้อความ (string)

```
char name[15] = "COMPUTER";    /* เก็บข้อความลงตัวแปร name */  
char phone[15] = "0-2737-3000";  
char age[3] = "20";
```

3.4 ประเภทของข้อมูลและตัวแปร

การประกาศค่าคงที่

1. ใช้คำสั่ง `const` ตามรูปแบบดังนี้

`const` ชนิดข้อมูล ชื่อตัวแปร = ค่าที่เก็บในตัวแปร;

```
const int count = 120 ;
```

2. ใช้ตัวประมวลผลก่อน ตามรูปแบบดังนี้

`#define` ชื่อค่าคงที่ ค่าคงที่ที่ต้องการเก็บ

```
#define count 120
```

3.4 ประเภทของข้อมูลและตัวแปร

การประกาศค่าคงที่

ตัวอย่าง const

```
#include<stdio.h>
int main()
{
    const double pi=3.14;
    const char ch= 'A';
    const char company[10]="INTER";
    printf("pi = %d\n",pi);
    printf("ch = %d\n",ch);
    printf("company name = %s",company);
    return 0;
}
```

ตัวอย่าง #define

```
#include<stdio.h>
#define PI 3.14
#define AREA(x) PI*x*x
int main()
{
    int r=2;
    printf("pi = %d\n",pi);
    printf("Area = %f",AREA(r) );
    return 0;
}
```


3.5 นิพจน์และตัวดำเนินการ

นิพจน์ (Expression) คือ การนำค่าคงที่หรือตัวแปรมาเชื่อมต่อกัน ด้วยเครื่องหมายทางคณิตศาสตร์

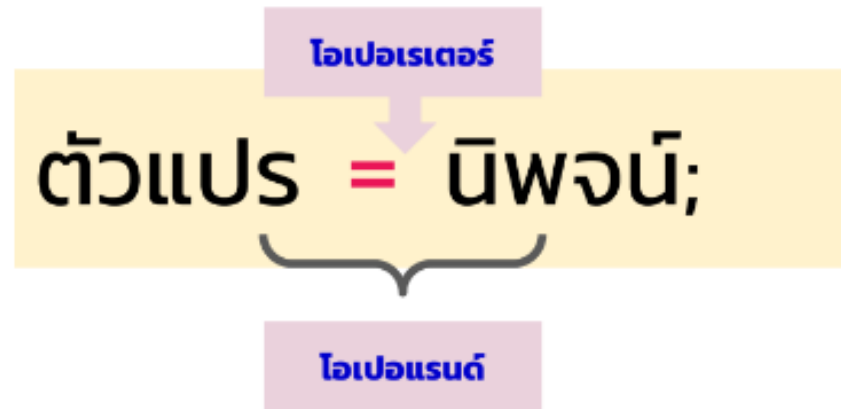
ตัวอย่าง

```
con = 10.5;
```

```
result = 25 * 6;
```

```
point = score1 * 2 + score2 * 5 + score3 * 8;
```

```
a = b = 0; /*ให้ตัวแปร a เท่ากับตัวแปร b โดยมีค่าเป็นศูนย์ */
```



3.5 นิพจน์และตัวดำเนินการ

```
1  #include<stdio.h>
2  int main(){
3      int sum, total;
4      sum = total = 0;
5      printf(" sum = %d \n",sum);
6      printf(" total = %d \n",total);
7
8      return 0;
9  }
10
```

sum = 0

total = 0

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการ (Operator) ในโปรแกรมภาษาซีมีตัวดำเนินการหลายชนิด
เพื่อใช้ในการเขียนโปรแกรม ดังนี้

- ตัวดำเนินการคณิตศาสตร์ (Arithmetic operators)
- ตัวดำเนินการกำหนดค่าแบบผสม (Compound assignment operators)
- ตัวดำเนินการยูนารี (Unary operators)
- ตัวดำเนินการเปรียบเทียบ (Comparison operators)
- ตัวดำเนินการตรรกะ (Logical operators)

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการคณิตศาสตร์ (Arithmetic operators)

การคำนวณ	ตัวดำเนินการ	ตัวอย่าง	การทำงาน
บวก	+	$c=a+b;$	นำ a บวก b ผลลัพธ์เก็บใน
ลบ	-	$c=a-b;$	นำ a ลบ b แล้วเก็บใน c
คูณ	*	$c=a*b;$	นำ a คูณ b แล้วเก็บใน c
หาร	/	$c=a/b$	นำ a หารด้วย b แล้วเก็บใน c
มอดุลัส	%	$c=a\%b;$	เป็นการหารที่เก็บเศษไว้ใน c

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการคณิตศาสตร์ (Arithmetic operators)

การหาแบบหารแบบเอาเศษหรือ Modulus %

$$9\%2 = ?$$

- ตั้งหารยาว 9 หาร 2
- ผลลัพธ์คือเศษจากการหาร

$$\begin{array}{r} 4 \\ 2 \overline{) 9} \\ \underline{8} \\ 1 \end{array}$$

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการคณิตศาสตร์ (Arithmetic operators)

ข้อควรระวัง !!!

สำหรับภาษาซี operand ที่จะนำมาดำเนินการกันผ่าน operator ใดๆ ต้องเป็น**ข้อมูลชนิดเดียวกัน** และผลลัพธ์ที่ได้ เป็นข้อมูลชนิดเดียวกันด้วย

เต็ม / เต็ม = เต็ม	→	$7 / 2 = 3$
เต็ม % เต็ม = เต็ม	→	$7 \% 2 = 1$
จริง / จริง = จริง	→	$7.0 / 2.0 = 3.5$

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการคณิตศาสตร์ (Arithmetic operators)

ข้อควรระวัง !!!

ถ้า a และ b ไม่ใช่ข้อมูลชนิดเดียวกัน ภาษาซีจะแปลงให้เป็นข้อมูลชนิดเดียวกันโดยอัตโนมัติก่อนคำนวณ

จริง / เต็ม = จริง $\longrightarrow 7.0 / 2.0 = 3.5$

เต็ม / จริง = จริง $\longrightarrow 7.0 / 2.0 = 3.5$

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการคณิตศาสตร์ (Arithmetic operators)

ข้อควรระวัง !!!

```
1 #include <stdio.h>
2 int main()
3 {
4     int num = 7/2;
5     printf("%d",num);
6     return 0;
7 }
```

ผลการรัน :

3

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการคณิตศาสตร์ (Arithmetic operators)

ข้อควรระวัง !!!

```
1 #include <stdio.h>
2 int main()
3 {
4     float num = 7/2;
5     printf("%f",num);
6     return 0;
7 }
```

ผลการรัน :

3.000000

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการกำหนดค่าแบบผสม (Compound assignment operators)

ประกอบด้วย $+=$, $-=$, $*=$, $/=$ และ $\% =$ โดยสามารถแสดงได้ดังนี้

นิพจน์ทั่วไป	นิพจน์แบบผสม
$a = a + 5$	$a += 5$
$a = a - 5$	$a -= 5$
$a = a * (b - 3)$	$a *= (b - 3)$
$a = a / 3$	$a /= 3$
$a = a \% (b - 2)$	$a \% = (b - 2)$

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการกำหนดค่าแบบผสม (Compound assignment operators)

เป็นการใช้ตัวดำเนินการกับตัวแปรตัวเดียว มีการใช้สองแบบคือ

1. ตัวดำเนินการเอกภาคเติมหน้า (prefix mode)

EX: $++a$; เพิ่มค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x+1$;))

$--a$; ลดค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x-1$;))

2. ตัวดำเนินการเอกภาคเติมหลัง (postfix mode)

EX: $a++$; เพิ่มค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x+1$;))

$a--$; ลดค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x-1$;))

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการกำหนดค่าแบบผสม (Compound assignment operators)

เป็นการใช้ตัวดำเนินการกับตัวแปรตัวเดียว มีการใช้สองแบบคือ

1. ตัวดำเนินการเอกภาคเติมหน้า (prefix mode)

EX: **++**a; เพิ่มค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x+1$;))

--a; ลดค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x-1$;))

2. ตัวดำเนินการเอกภาคเติมหลัง (postfix mode)

EX: a**++**; เพิ่มค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x+1$;))

a**--**; ลดค่าครึ่งละหนึ่งค่า (มีค่าเท่ากับ $x=x-1$;))

การเพิ่มหรือลดค่าไว้
หน้าและหลังตัวแปร
จะได้ผลไม่เหมือนกัน

3.5 นิพจน์และตัวดำเนินการ

ตัวดำเนินการกำหนดค่าแบบผสม (Compound assignment operators)

$y = x++$ หมายถึง จะส่งค่า x ให้ y ก่อน จากนั้นเพิ่มค่าให้ตัวมันเองหนึ่งค่า

$y = ++x$ หมายถึง จะเพิ่มค่าให้ตัวมันเองหนึ่งค่า แล้วค่อยส่งค่า x ให้ y

ถ้าหากสมมติว่า $i = 10$ และ $j = 2$ พิจารณาการดำเนินการต่อไปนี้

$i = i * j++$ ผลลัพธ์จะเท่ากับ $i = 20$ และ $j = 3$ (เพิ่มค่าหลังจากการคูณ)

$i = i * ++j$ ผลลัพธ์จะเท่ากับ $i = 30$ และ $j = 3$ (เพิ่มค่าก่อนการคูณ)

$i = i * j--$ ผลลัพธ์จะเท่ากับ $i = 20$ และ $j = 1$ (ลดค่าหลังการคูณ)

$i = i * --j$ ผลลัพธ์จะเท่ากับ $i = 10$ และ $j = 1$ (ลดค่าก่อนการคูณ)

3.5 นิพจน์และตัวดำเนินการ

ลำดับการคำนวณ

ลำดับที่	ตัวดำเนินการ
1	()
2	++ --
3	* / %
4	+ -
5	+= *= /= -= %=

ซ้าย -> ขวา



3.5 นิพจน์และตัวดำเนินการ

ลำดับการคำนวณ

$$3 * (4 \% (6 / 2)) + 5$$

$$3 * (4 \% 3) + 5$$

$$3 * 1 + 5$$

8

ทำในวงเล็บในก่อน
ทำวงเล็บนอก

3.5 นิพจน์และตัวดำเนินการ

ลำดับการคำนวณ

$$\begin{array}{ccccccc} 5 & \% & 2 & + & 14 & / & 7 & - & 1 \\ \underbrace{\hspace{1.5cm}} & & & & \underbrace{\hspace{1.5cm}} & & & & \\ 1 & & + & & 2 & & - & 1 \\ \underbrace{\hspace{2.5cm}} & & & & & & & & \\ 3 & & & & & & - & 1 \\ \underbrace{\hspace{3.5cm}} & & & & & & & & \\ 2 & & & & & & & & \end{array}$$

3.5 นิพจน์และตัวดำเนินการ

ลำดับการคำนวณ

ผลลัพธ์เป็นเท่าไร

$$2.0/2.0+5*2 = 11.0$$

$$2.0*2.0+5*2 = 14.0$$

$$2.0+5/2.0*2 = 7.0$$

3.6 คำสั่งรับข้อมูล

- `scanf ()` รับข้อมูลข้อความจากคีย์บอร์ดมาเก็บในตัวแปรที่กำหนด
- `gets()` รับข้อความ (string) มาเก็บในตัวแปรที่กำหนด
- `getchar()` รับข้อมูลอักขระตัวเดียว และต้องกด Enter ทุกครั้ง
- `getch()` รับข้อมูลอักขระตัวเดียว ไม่ต้องกด Enter และไม่แสดงผล
- `getche()` รับข้อมูลอักขระตัวเดียวเหมือน `getch()` แต่จะแสดงผลด้วย

3.6 คำสั่งรับข้อมูล

คำสั่ง scanf()

รูปแบบคำสั่ง

```
scanf("control string", argument list);
```

ต้อง include<stdio.h>

ตัวอย่างการใช้งาน

```
int score;  
printf("Enter score :");  
scanf("%d", &score);
```

argument list ตัวแปรที่ใช้เก็บข้อมูล ใช้เครื่องหมาย & นำหน้าชื่อ แต่ถ้ารับข้อมูลเป็นสตริงไม่ต้องมีเครื่องหมายนี้

control string เป็นรหัสรูปแบบข้อมูล (format code)

หากป้อนข้อมูลแล้ว**เว้นวรรค** โปรแกรมจะมองว่าเป็น**ตัวแปรมากกว่าหนึ่งตัว**

3.6 คำสั่งรับข้อมูล

คำสั่ง scanf()

ตัวอย่างโปรแกรมคำนวณพื้นที่วงกลม

```
1 #include<stdio.h>
2 int main(){
3     float r,area;
4     printf("Enter radius:");
5     scanf("%f",&r);
6     area=(22.0/7.0)*r*r;
7     printf("%.2f",area);
8 }
```

format code

%c

ใช้แสดงอักขระตัวเดียว (Char)

%s

ใช้แสดงชุดตัวอักษรหรือข้อความ (String)

%d

ใช้แสดงจำนวนเต็ม (Integer)

%f

ใช้แสดงเลขทศนิยม (Float)

%h

ใช้แสดง short integer

%u

ใช้แสดง unsigned int

%o

ใช้แสดงจำนวนเต็มเลขฐาน 8

%x

ใช้แสดงข้อมูลเลขฐาน 16

3.6 คำสั่งรับข้อมูล

คำสั่ง gets()

รูปแบบคำสั่ง

```
gets (stringVar)
```

ตัวอย่างการใช้งาน

```
char stringVar[5];  
gets (stringVar);
```

เป็นตัวแปรสตริงที่ใช้เก็บข้อความ (string constant)
ฟังก์ชันนี้ใช้รับข้อมูลจากคีย์บอร์ดมาเก็บในตัวแปรสตริง

สามารถรับข้อมูลที่มี เว้นวรรค ได้

stringVar เป็นตัวแปรสตริงที่ใช้เก็บ
ข้อความ (string constant)

3.6 คำสั่งรับข้อมูล

คำสั่ง gets()

```
1  #include <stdio.h>
2  int main(){
3      char name[30];
4      int age;
5      printf("Enter your name :");
6      gets(name);
7      printf("Enter your age :");
8      scanf("%d",&age);
9      printf("Your name is %s\n",name);
10     printf("Your are %d years old. \n",age);
11     return 0;
12 }
```

3.6 คำสั่งรับข้อมูล

คำสั่ง getchar()

รูปแบบคำสั่ง

```
getchar()
```

ตัวอย่างการใช้งาน

```
char charVar;  
charVar = getchar();
```

charVar เป็นตัวแปรชนิด char
ที่ใช้เก็บข้อมูลตัวอักษรที่ป้อน
ผ่านคีย์บอร์ด

3.6 คำสั่งรับข้อมูล

คำสั่ง getchar()

```
1 #include <stdio.h>
2 int main(){
3     char c;
4     printf("Enter a single character :");
5     c = getchar( );
6     printf("You type a character is %c\n",c);
7     return 0;
8 }
```


3.6 คำสั่งรับข้อมูล

คำสั่ง getch()

ต้อง `include<conio.h>`

รูปแบบคำสั่ง

`getch ()`

ฟังก์ชันนี้ใช้รับอักขระตัวเดียวจากคีย์บอร์ด
โดยไม่ต้องกด Enter ข้อมูลที่รับเข้าไปจะไม่แสดงผล

ตัวอย่างการใช้งาน

```
char charVar;  
charVar = getch ();
```

charVar เป็นตัวแปรชนิด char
ที่ใช้เก็บข้อมูลตัวอักขระที่ป้อน
ผ่านคีย์บอร์ด

3.6 คำสั่งรับข้อมูล

คำสั่ง getch()

```
1 #include <stdio.h>
2 #include <conio.h>
3 int main(){
4     char d;
5     printf("Enter a single character :");
6     d = getch();
7     printf("You type a character is ... %c\n",d);
8     return 0;
9 }
```

3.6 คำสั่งรับข้อมูล

คำสั่ง getch()

สรุปการใช้งาน

- รับข้อมูล string ควรใช้ฟังก์ชัน gets() หรือ scanf()
- รับข้อมูลตัวเลข ควรใช้ฟังก์ชัน scanf() ถ้าหากป้อนข้อมูลแล้วเว้นวรรค โปรแกรมจะมองว่าเป็นตัวแปรมากกว่าหนึ่งตัว
- รับตัวเลขหนึ่งตัวหรือตัวอักขระหนึ่งตัว ไม่แสดงทางจอภาพ ใช้ getch() แต่ถ้าต้องการให้แสดงทางจอภาพใช้ getchar()

แบบฝึกหัด

- 1.1) จงเขียนโปรแกรมคำนวณพื้นที่สามเหลี่ยม กำหนดให้ค่าเริ่มต้น สูง=10 หน่วย ฐาน=14 หน่วย
- 1.2) จงเขียนโปรแกรมคำนวณพื้นที่สามเหลี่ยม โดยรับค่าความยาวฐานและความสูงจากคีย์บอร์ด
- 1.3) จงเขียนโปรแกรมคำนวณหาค่า y จากสมการ $y = x^2 + 8x + 4$ โดยให้รับค่า x จากคีย์บอร์ด
- 1.4) จงเขียนโปรแกรมรับค่าปี พ.ศ. จากคีย์บอร์ด แล้วแสดงผลเป็น ค.ศ.
- 1.5) จงเขียนโปรแกรมรับปี จากคีย์บอร์ด หลังจากนั้นให้ทำการแปลงเลขปี ค.ศ. ที่รับเข้ามานั้นให้กลายเป็นปีพ.ศ. หลังจากนั้นให้ทำการคำนวณผลรวมของเลขแต่ละหลักของปี ค.ศ. และ พ.ศ. ที่ได้มานั้น แสดงผลลัพธ์สุดท้ายออกทางจอภาพ ดังต่อไปนี้

แบบฝึกหัด

1.6) จงเขียนโปรแกรมวัดความเหมาะสมระหว่างส่วนสูงกับน้ำหนักด้วยการหาค่าดัชนีมวลกาย (Body Mass Index : BMI) ตามสูตรต่อไปนี้

$$BMI = \frac{Weight (kg)}{Height^2 (m^2)}$$

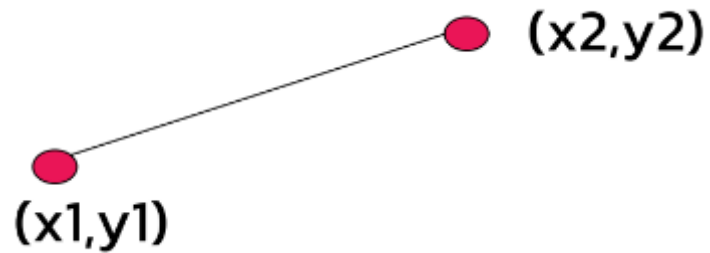
1.7) จงเขียนโปรแกรมสำหรับการกด ATM โดยป้อนตัวเลขเข้าไป แล้วให้จำนวนแบงก์ออกมา เช่น

INPUT	4800
OUTPUT	B1000 = 4
	B500 = 1
	B100 = 3

แบบฝึกหัด

1.8) จงเขียนโปรแกรมหาระยะระหว่างจุดสองจุด อินพุตเป็น (x_1, y_1) และ (x_2, y_2)

หมายเหตุ : `#include<math.h>` เพื่อใช้ฟังก์ชัน `sqrt()`



ฝึกฝนเพิ่มเติมได้ที่

<https://www.programming.in.th/>