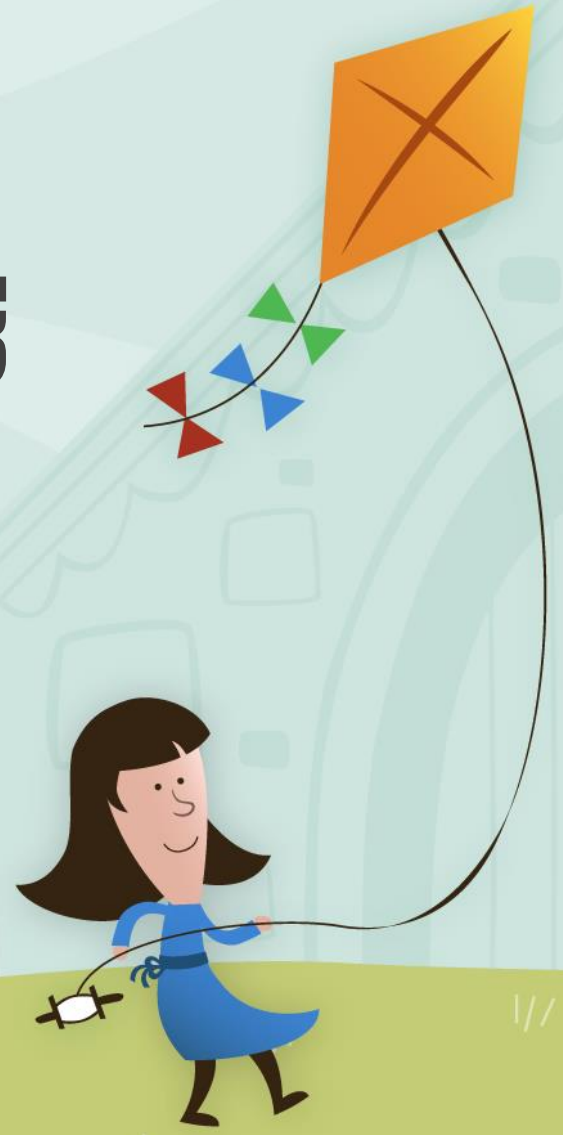
Introduction To C Language

ความรู้เบื้องต้นเกี่ยวกับภาษาซี

ห้องเรียนพิเศษวิทยาศาสตร์ คณิตศาสตร์ เทคโนโลยีและสิ่งแวดล้อม (SMTE)

ชั้นมัธยมศึกษาปีที่ 4



ครูผู้สอน นายวรเทพ วันกาล

โรงเรียนนารีรัตน์จังหวัดแพร่

ความสำคัญ

ภาษาซีจัดเป็นภาษาระดับกลางที่มีลักษณะเป็นภาษาโครงสร้างสามารถประยุกต์ใช้ได้กับงานในลักษณะต่างๆ เป็นภาษาที่ใกล้เคียงกับภาษาแอสแซมบลี ผู้เขียนโปรแกรมจะสามารถเขียนโปรแกรมได้อย่างคล่องตัวโดยไม่มีข้อจำกัดในการวางตำแหน่งฟังก์ชันในโปรแกรม ภาษาซีจึงเป็นภาษาที่ง่ายต่อการเข้าใจและการนำไปใช้งาน การสร้างโปรแกรมภาษาซีจะเริ่มจากการเขียนโปรแกรมต้นกำเนิด แล้วนำไปทำการแปลด้วยตัวแปลภาษาซีเกิดเป็นโปรแกรมประสงค์ หลังจากนั้น จึงนำโปรแกรมประสงค์ไปทำการเชื่อมโยง เพื่อให้เกิดเป็นโปรแกรมทำการที่สามารถทำงานได้อย่างรวดเร็ว

Dennis Ritchie ผู้คิดค้นสร้างภาษาซีขึ้นเป็นคนแรกในปี 1972

ขั้นตอนการพัฒนาโปรแกรมภาษาซี

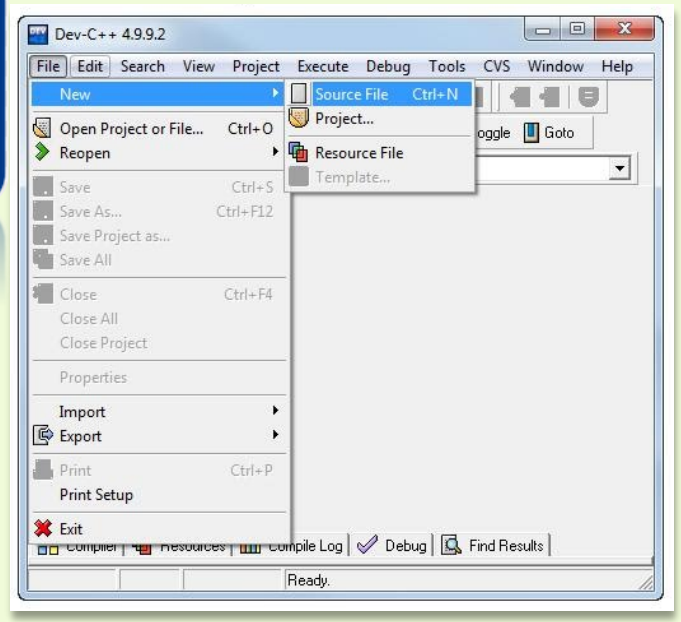
ขั้นตอนที่ 1 เขียนโปรแกรม (Source Code)

ขั้นตอนที่ 2 คอมไพล์โปรแกรม (Compile)

ขั้นตอนที่ 3 เชื่อมโยงโปรแกรม (Link)

ขั้นตอนที่ 4 ประมวลผล (Run)

ขั้นตอนที่ 1 เขียนโปรแกรม (Source Code)



ใช้ Editor เขียนโปรแกรมภาษาซีและทำการบันทึกไฟล์ให้มีนามสกุลเป็น .c เช่น work.c เป็นต้น Editor คือ โปรแกรมที่ใช้สำหรับการเขียนโปรแกรม โดยตัวอย่างของ Editor ที่นิยมนำมาใช้ในการเขียนโปรแกรมได้แก่ Notepad, Edit ของ Dos, TextPad และ EditPlus เป็นต้นการเขียนโปรแกรมสามารถเลือกใช้โปรแกรมใดในการเรียนโปรแกรมก็ได้ แล้วแต่ความถนัดของแต่ละบุคคล

ขั้นตอนที่ 2 คอมไพล์โปรแกรม (Compile)

นำ Source Code จากขั้นตอนที่ 1 มาทำการคอมไพล์ เพื่อแปลจากภาษาซีที่มนุษย์เข้าใจไปเป็นภาษาเครื่องที่คอมพิวเตอร์เข้าใจได้ ในขั้นตอนนี้คอมไพเลอร์จะทำการตรวจสอบ Source Code ว่าเกิดข้อผิดพลาดหรือไม่

หากเกิดข้อผิดพลาด จะแจ้งให้ผู้เขียนโปรแกรมทราบ ผู้เขียนโปรแกรมจะต้องกลับไปแก้ไขโปรแกรมและทำการคอมไพล์โปรแกรมใหม่อีกครั้ง

หากไม่พบข้อผิดพลาด คอมไพเลอร์จะแปลไฟล์ Source Code จากภาษาซีไปเป็นภาษาเครื่อง (ไฟล์นามสกุล .obj) เช่นถ้าไฟล์ Source Code ชื่อ work.c ก็จะถูกแปลไปเป็นไฟล์ work.obj ซึ่งเก็บภาษาเครื่องไว้เป็นต้น

Compile เป็นตัวแปลภาษารูปแบบหนึ่ง มีหน้าที่หลักคือการแปลภาษาโปรแกรมที่มนุษย์เขียนขึ้นไปเป็นภาษาเครื่อง โดยคอมไพเลอร์ของภาษาซี คือ C Compiler ซึ่งหลักการที่คอมไพเลอร์ใช้ เรียกว่า คอมไพล์ (Compile) โดยจะทำการอ่านโปรแกรมภาษาซีทั้งหมดตั้งแต่ต้นจนจบ แล้วทำการแปลผลทีเดียว

ขั้นตอนที่ 2 คอมไพล์โปรแกรม (Compile)

คอมไพเลอร์ (compiler) เป็นตัวแปลภาษาอีกรูปแบบหนึ่ง มีหน้าที่หลักคือการแปลภาษาโปรแกรมที่มนุษย์เขียนไปเป็นภาษาเครื่อง โดยจะทำการอ่านโปรแกรมภาษาซีทั้งหมดตั้งแต่ต้นจนจบแล้วทำการแปลผลทีเดียว

อินเตอร์พรีเตอร์ (interpreter) ก็เป็นตัวแปลภาษาอีกรูปแบบหนึ่งเหมือนกัน แต่หลักการทำงานจะแตกต่างกันออกไปคือ อินเตอร์พรีเตอร์จะทำการอ่านและแปลโปรแกรมที่ละบันทัด เมื่อแปลผลบรรทัดหนึ่ง เสร็จก็จะทำตามคำสั่ง ในบรรทัดนั้น แล้วจึงทำการแปลผลและทำตามคำสั่งในบรรทัดถัดไป

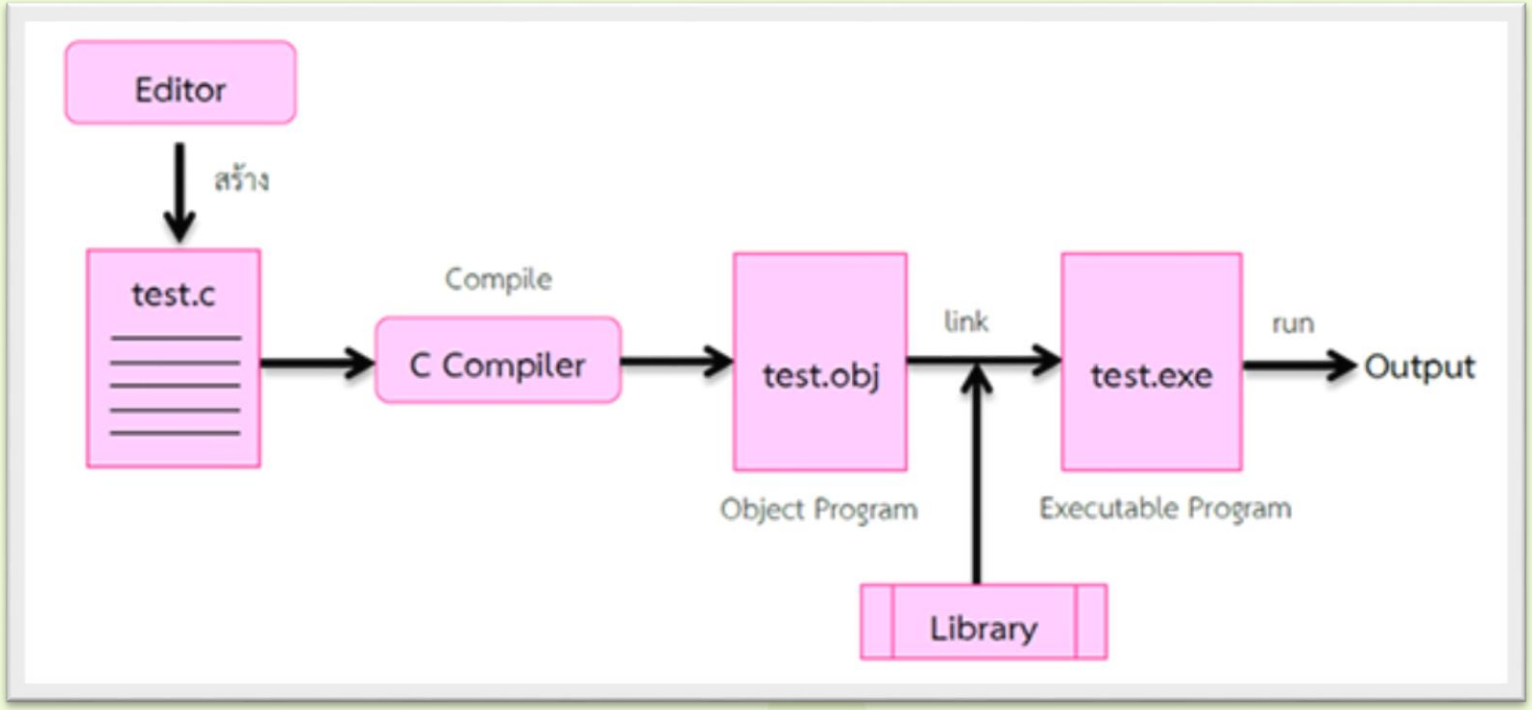
ขั้นตอนที่ 3 เชื่อมโยงโปรแกรม (Link)

การเขียนโปรแกรมภาษาซีนั้นผู้เขียนโปรแกรมไม่จำเป็นต้องเขียนคำสั่งต่าง ๆ ขึ้นใช้งานเอง เนื่องจาก ภาษาซีมีฟังก์ชันมาตรฐานให้ผู้เขียนโปรแกรมสามารถเรียกใช้งานได้ เช่น การเขียนโปรแกรมแสดงข้อความ “Nareerat" ออกทางหน้าจอ ผู้เขียนโปรแกรมสามารถเรียกใช้ฟังก์ชัน printf() ซึ่งเป็นฟังก์ชัน มาตรฐานของภาษาซีมาใช้งานได้ โดยส่วนการประกาศ (Declaration) ของฟังก์ชั่นมาตรฐานต่าง ๆ จะถูกจัดเก็บอยู่ในเฮดเดอร์ไฟล์แต่ละตัว แตกต่างกันไปตามลักษณะการใช้งาน

ด้วยเหตุนี้ภาษาเครื่องที่ได้จากขั้นตอนที่ 2 จึงยังไม่สามารถนำไปใช้งานได้ แต่ต้องนำมาเชื่อมโยงเข้ากับ Library ก่อน ซึ่งผลจากการเชื่อมโยงจะทำให้ได้ Executable Program (ไฟล์นามสกุล.exe เช่น work.exe) ที่สามารถนำไปใช้งานได้

ขั้นตอนที่ 4 ประมวลผล (Run)

เมื่อนำ Executable Program จากขั้นตอนที่ 3 มาประมวลผลก็จะได้ผลลัพธ์ (Output) ของโปรแกรมออกมา



แนวคิดในการเขียนโปรแกรมคอมพิวเตอร์

ในการพัฒนาโปรแกรมมีขั้นตอนหลัก 5 ขั้นตอน

1. วิเคราะห์ปัญหา (Analysis)
2. วางแผนและออกแบบ (Planning & Design)
3. เขียนโปรแกรม (Coding)
4. ทดสอบโปรแกรม (Testing)
5. จัดทำคู่มือ (Documentation)

1. วิเคราะห์ปัญหา (Analysis)

ขั้นตอนนี้ถือว่าเป็นขั้นตอนที่สำคัญที่สุด ผู้เขียนโปรแกรมต้องวิเคราะห์ปัญหาให้ออกว่าจะต้องทำการเขียนโปรแกรมเพื่อแก้ปัญหาอะไร เพราะหากวิเคราะห์หรือมองปัญหาผิดแล้ว ก็จะทำให้เขียนโปรแกรมได้ผลลัพธ์ออกมาผิดไปจากสิ่งที่ต้องการด้วย และนอกจากจะวิเคราะห์ว่าปัญหาคืออะไรแล้ว จำเป็นอย่างยิ่งที่จะต้องวิเคราะห์ด้วยว่าข้อมูลที่จะนำเข้ามาใช้ในโปรแกรมมีอะไรบ้าง

1. วิเคราะห์ปัญหา (Analysis)

โจทย์ : จงเขียนโปรแกรมรับค่าเลขจำนวนเต็ม 2 จำนวน และหาผลบวกของเลขทั้ง 2 จำนวนนั้น

จากโจทย์ข้างต้น สามารถแตกปัญหาได้เป็น 2 ส่วน คือ

> ต้องรับข้อมูลเลขจำนวนเต็ม 2 ตัวเข้ามาในโปรแกรม

วิเคราะห์ :: กำหนดให้ X เก็บเลขจำนวนเต็มตัวที่ 1 กำหนดให้ y เก็บเลขจำนวนเต็มตัวที่ 2

> เลขจำนวนเต็มตัวที่ 1 + เลขจำนวนเต็มตัวที่ 2 มีค่าเท่ากับเท่าไร

วิเคราะห์ :: กำหนดให้ Sum เก็บค่าผลบวกของเลขจำนวนเต็มทั้ง 2 จำนวน นั่นคือ Sum = X+Y

2. วางแผนและออกแบบ (Planning & Design)

การวางแผน คือ การนำปัญหาที่วิเคราะห์ได้จากขั้นตอนที่ 1 มาวางแผนอย่างเป็นขั้นตอน จะต้องเขียนโปรแกรมเพื่อแก้ปัญหาอย่างไร การวางแผนอย่างเป็นขั้นตอนนี้ เรียกว่า อัลกอริต (Algorithm) ซึ่งอัลกอริทึมแบ่งออกเป็น 2 รูปแบบ คือ

ซูโดโค้ด (Pseudo Code)

โฟลวชาร์ต (Flowchart)

2. วางแผนและออกแบบ (Planning & Design)

ซูโดโค้ด (Pseudo Code)

ซูโดโค้ด (Pseudo Code) คือ การเขียนอัลกอริทึม โดยใช้ประโยคภาษาอังกฤษที่สื่อความหมายง่าย ๆ สามารถอ่านแล้วเข้าใจได้โดยทันที

รูปแบบ

Algorithm <ชื่อของอัลกอริทึม>
1.....
2.....
.....
END

2. วางแผนและออกแบบ (Planning & Design)

ซูโดโค้ด (Pseudo Code)

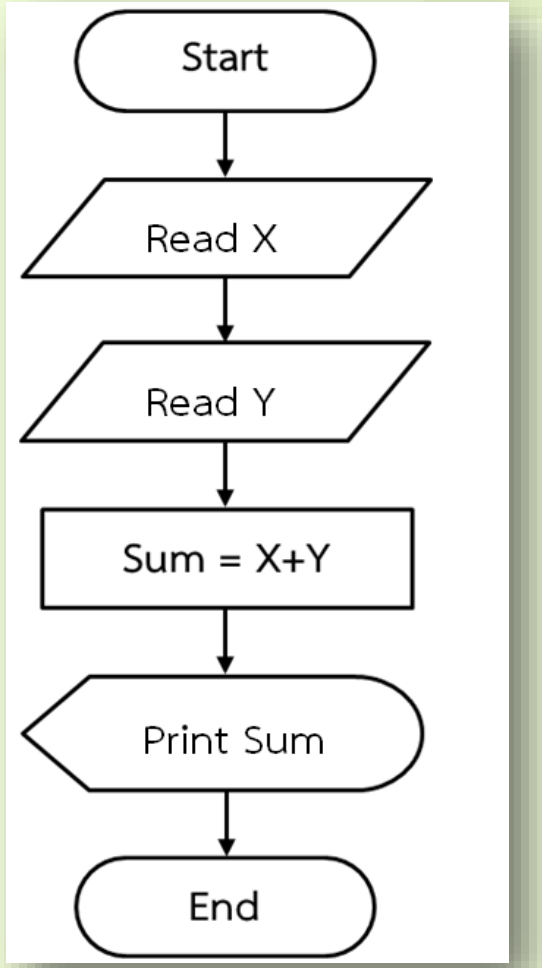
การเขียนซูโดโค้ด คำนวณหาผลบวกของเลขทั้ง 2 จำนวน สามารถเขียนเป็นภาษาไทยหรือเขียนเป็นภาษาอังกฤษได้ดังนี้

การเขียนซูโดโค้ดภาษาไทย	การเขียนซูโดโค้ดภาษาอังกฤษ
Algorithm การหาผลบวกของเลขทั้ง 2 จำนวน 1. เริ่มต้น 2. รับค่าจำนวนเต็มตัวที่ 1 3. รับค่าจำนวนเต็มตัวที่ 2 4. คำนวณหาผลบวกของเลขทั้ง 2 จำนวน 5. แสดง ผลบวกของเลขทั้ง 2 จำนวน 6. จบ	Algorithm Average_Sum 1. Start 2. READ X 3. READ Y 4. Sum = X+Y 5. Print Sum 6. End

2. วางแผนและออกแบบ (Planning & Design)

โฟลวชาร์ต (Flowchart)

โฟลวชาร์ต (Flowchart) คือ การเขียนอัลกอริทึม โดยใช้สัญลักษณ์รูปภาพเป็นตัวสื่อความหมาย จากโจทย์สามารถเขียนโฟลวชาร์ตได้ดังรูป



3. เขียนโปรแกรม (Coding)

บรรทัดที่	ซอร์สโค้ด
1:	#include <stdio.h>
2:	void main(void)
3:	{
4:	int X , Y , Sum;
5:	printf("READ X is : ");
6:	scanf("%d",&X);
7:	printf("READ Y is : ");
8:	scanf("%d",&Y);
9:	Sum = X + Y;
10	printf("Sum of %d + %d is
11:	%d\n",X,Y,Sum);
	}

เป็นการนำอัลกอริทึมจากขั้นตอนที่ 2 มาเขียนโปรแกรมให้ถูกต้องตามหลักไวยากรณ์ (Syntax) ของภาษาซี จากโจทย์สามารถเขียนโปรแกรมได้ดังนี้

3. เขียนโปรแกรม (Coding)

หากนำโปรแกรม (Source Code) มาพิจารณา จะพบว่าการเขียนโปรแกรมมีขั้นตอนเป็นไปตามขั้นตอนของอัลกอริทึมที่ได้วิเคราะห์ขึ้นทุกประการ ดังนี้

บรรทัดที่	ซอร์สโค้ด	อัลกอริทึม
1:	#include <stdio.h>	
2:	void main(void)	
3:	{	
4:	int X , Y , sum;	INPUT X
5:	printf("READ X is : "); ---------->	
6:	scanf("%d",&X);	INPUT Y
7:	printf("READ Y is : "); ---------->	
8:	scanf("%d",&Y);	SUM = X+Y
9:	Sum = X + Y; ---------->	PRINT SUM
10	printf("Sum of %d + %d is %d\n",X,Y,Sum); ->	
11:	}	

4. ทดสอบโปรแกรม (Testing)

เป็นการนำผลลัพธ์จากขั้นตอนที่ 3 มาทำการรัน (Run) โดยทดสอบป้อนค่า X และ Y เข้าไปในโปรแกรม และตรวจสอบผลลัพธ์ที่ได้ว่าถูกต้องหรือไม่ ให้ทดสอบหลาย ๆ ครั้ง หากผลลัพธ์ที่ได้ถูกต้อง แสดงว่าโปรแกรมที่เขียนขึ้นถูกต้องแล้ว แต่หากผลลัพธ์ถูกบ้างผิดบ้างหรือผิดทุกครั้ง แสดงว่าโปรแกรมที่เขียนขึ้นผิดพลาด ผู้เขียนโปรแกรมต้องกลับไปตรวจสอบ และแก้ไขโปรแกรมใหม่อีกครั้ง

รันครั้งที่ 1
READ X is : 7
READ Y is : 8
Sum of 7 + 8 is 15

รันครั้งที่ 2
READ X is : 23
READ Y is : 37
Sum of 23 + 37 is 60

รันครั้งที่ 3
READ X is : 51
READ Y is : 60
Sum of 51 + 60 is 111

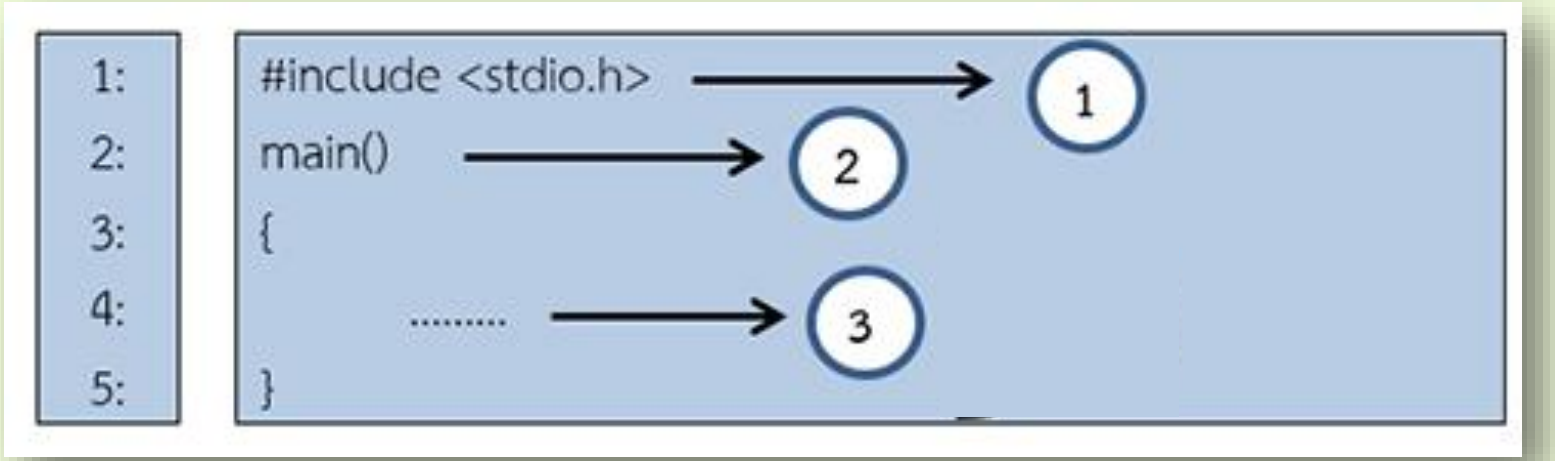
5. จัดทำคู่มือ (Documentation)

จุดประสงค์ที่สำคัญของการทำคู่มือ คือ ช่วยให้ผู้อื่นศึกษาซอร์สโค้ด (Source Code) ของโปรแกรมได้ง่ายขึ้น ซึ่งจะเป็นประโยชน์มากสำหรับการพัฒนาโปรแกรมในอนาคต เพราะจะช่วยให้ศึกษาซอร์สโค้ดได้ง่ายและรวดเร็วขึ้น การจัดทำคู่มือไม่มีกฎเกณฑ์ระบุไว้แน่นอน แต่ผู้เขียนโปรแกรมควรจัดทำคู่มือให้มีรายละเอียดมากที่สุด จากโจทย์ สามารถจัดทำคู่มือได้ดังนี้ (การจัดทำคู่มือที่จะแสดงต่อไปนี้ เป็นเพียงตัวอย่าง ผู้อ่านสามารถจัดทำคู่มือออกมาในลักษณะอื่น ๆ ได้ตามที่ต้องการ แต่สิ่งสำคัญที่ลืมไม่ได้ คือ ควรจัดทำคู่มือให้รายละเอียดมากที่สุด)

ชื่อโปรแกรม	การหาค่าผลบวกของเลขจำนวนเต็ม 2 จำนวน
ตัวแปรที่ใช้	X เก็บค่าจำนวนเต็มตัวที่ 1 Y เก็บค่าจำนวนเต็มตัวที่ 2 Sum เก็บค่าผลบวกของเลขจำนวนเต็มทั้ง 2 จำนวน
ชนิดของข้อมูล	X, Y, Sum เป็นข้อมูลชนิดเลขจำนวนเต็ม (Integer)
วิธีการแก้ปัญหา	ใช้สมการ Sum = X + Y

โครงสร้างของภาษาภาษาซี

โครงสร้างของภาษาภาษาซี



โครงสร้างของภาษาภาษาซีแบ่งออกเป็น 3 ส่วนดังนี้

1. ส่วนหัวของโปรแกรม
2. ส่วนของฟังก์ชันหลัก
3. ส่วนรายละเอียดของโปรแกรม

โครงสร้างของภาษาภาษาซี

1 ส่วนหัวของโปรแกรม

ส่วนหัวของโปรแกรมนี้เรียกว่า Preprocessing Directive ใช้ระบุเพื่อบอกให้คอมไพเลอร์กระทำการใด ๆ ก่อนการแปลผลโปรแกรมในที่นี่คำสั่ง #include <stdio.h> ใช้บอกกับคอมไพเลอร์ให้นำเฮดเดอร์ไฟล์ที่ระบุ คือ stdio.h เข้าร่วมในการแปลโปรแกรมด้วย โดยการกำหนด Preprocessing Directives นี้จะต้องขึ้นต้นด้วยเครื่องหมาย # เสมอ

คำสั่งที่ใช้ระบุให้คอมไพเลอร์นำเฮดเดอร์ไฟล์เข้าร่วมในการแปลโปรแกรมสามารถเขียนได้ 2 รูปแบบ คือ

#include <ชื่อเฮดเดอร์ไฟล์> คอมไพเลอร์จะทำการค้นหาเฮดเดอร์ไฟล์ที่ระบุจากไดเรกทอรีที่ใช้ สำหรับเก็บเฮดเดอร์ไฟล์โดยเฉพาะ (ปกติคือไดเรกทอรีชื่อ include)

#include “ชื่อเฮดเดอร์ไฟล์” คอมไพเลอร์จะทำการค้นหาเฮดเดอร์ไฟล์ที่ระบุจากไดเร็คทอรีเดียวกันกับไฟล์ Source Code นั้น แต่ถ้าไม่พบก็จะไปค้นหาไดเร็คทอรีที่ใช้เก็บเฮดเดอร์ไฟล์โดยเฉพาะ

โครงสร้างของภาษาภาษาซี

2 ส่วนของฟังก์ชันหลัก

ฟังก์ชันที่กำหนดขึ้นมาชื่อฟังก์ชัน main() โดยทุกโปรแกรมจะต้องมีฟังก์ชัน main() ทำหน้าที่เป็นฟังก์ชันหลักในการทำงานในการประมวลผลโปรแกรมทุกครั้ง

ฟังก์ชันหลักของภาษาซี คือ ฟังก์ชัน main() ซึ่งโปรแกรมภาษาซีทุกโปรแกรมจะต้องมีฟังก์ชันนี้อยู่ในโปรแกรมเสมอ จะเห็นได้จากชื่อฟังก์ชันคือ main แปลว่า “หลัก” ดังนั้น การเขียนโปรแกรมภาษาซีจึงขาดฟังก์ชันนี้ไปไม่ได้ โดยขอบเขตของฟังก์ชันจะถูกกำหนดด้วยเครื่องหมาย { และ } กล่าวคือ การทำงานของฟังก์ชันจะเริ่มต้นที่เครื่องหมาย { และจะสิ้นสุดที่เครื่องหมาย } ฟังก์ชัน main() สามารถเขียนในรูปแบบของ int main ก็ได้ มีความหมายเหมือนกัน คือ หมายความว่า ฟังก์ชัน main() จะไม่มีอาร์กิวเมนต์ (Argument) คือไม่มีการรับค่าใด ๆ เข้ามาประมวลผลภายในฟังก์ชัน และจะมีการคืนค่ากลับออกไปจากฟังก์ชันด้วย

โครงสร้างของภาษาภาษาซี

2 ส่วนของฟังก์ชันหลัก

main() เทียบเท่ากับ void main(void) ----> ไม่คืนค่าใด ๆ กลับออกไปจากฟังก์ชัน

Argument คือ ตัวรับค่าเข้ามาในฟังก์ชัน

Parameter คือ ค่าที่ส่งไปยังฟังก์ชันค่า Argument และ Parameter ต้องเป็นข้อมูลชนิดเดียวกัน เช่น หากกำหนดให้ Argument เป็นข้อมูลชนิดตัวอักษรแล้วค่า Parameter ที่ส่งไปก็ต้องเป็นชนิดตัวอักษรด้วย

โครงสร้างของภาษาภาษาซี

2 ส่วนของฟังก์ชันหลัก

ตัวอย่าง argument และ parameter

```
1:      #include <stdio.h>
2:      void show (char a) ----------> Argument รับตัวอักษร 'a' มาในฟังก์ชัน
3:      {
4:          printf("%c",a) ;
5:      }
6:      void main(void) Parameter ส่งตัวอักษร 'a' ไปยังฟังก์ชัน show( )
7:      {
8:          show('a') ;
9:      }
```

โครงสร้างของภาษาภาษาซี

3 ส่วนรายละเอียดของโปรแกรม

เป็นส่วนของการเขียนคำสั่งต่าง ๆ เพื่อสั่งให้คอมพิวเตอร์ทำงาน ในการเขียนคำสั่งจะเขียนภายในเครื่องหมายปีกกาเปิด { และเครื่องหมายปีกกาปิด } โดยปกติส่วนของการเขียนโปรแกรมจะสามารถแบ่งออกได้เป็น 2 ส่วนด้วยกัน คือ

1) ส่วนของการประกาศตัวแปร คือ ส่วนที่ใช้ในการกำหนดตัวแปรที่จะใช้งานในการเขียนโปรแกรม

2) ส่วนของคำสั่ง หรือ ฟังก์ชันต่าง ๆ คือ ส่วนที่ใช้ในการกำหนดตัวแปรที่จะใช้งานในการเขียนโปรแกรม พิมพ์ฟังก์ชันเสร็จแล้วจะต้องปิดท้ายด้วยเครื่องหมายเซมิโคลอน ; เสมอ

คอมเมนต์ในภาษาซี

Comment เป็นส่วนของโปรแกรมที่คอมไพเลอร์ไม่ต้องคอมไพล์ หมายความว่า เมื่อคอมไพเลอร์ทำการคอมไพล์มาถึงส่วนของ Comment คอมไพเลอร์จะข้ามส่วนนั้นไปโดยไม่ต้องคอมไพล์

คอมเมนต์ในภาษาซีมี 2 แบบ

1. Line Comment

2. Block Comment

คอมเมนต์ในภาษาซี

1. Line Comment

เป็น Comment บรรทัดเดียว ใช้เครื่องหมาย // ซึ่ง Comment ประเภทนี้จะมีผลต่อบรรทัดหรือข้อความที่อยู่หลัง เครื่องหมาย // เพียงบรรทัดเดียวเท่านั้น ดังนั้นถ้าต้องการทำ comment หลาย ๆ บรรทัด จึงต้องเขียน // ในทุก ๆ บรรทัดที่ทำ Comment เช่น

```
// บรรทัดนี้เป็น Comment หรือหมายเหตุ หรือคำอธิบายโปรแกรม
// Comment จะไม่มีผลต่อโปรแกรม
// Comment ประเภทนี้ มีผลเพียงบรรทัดเดียวเท่านั้น
// Comment จะวางไว้ตรงไหนของโปรแกรมก็ได้
```

```c
#include <stdio.h>

int main(void)
{

    // This is
    // block
    // comment
    printf("Hello world\n");

    return 0;
}
```

คอมเมนต์ในภาษาซี

2. Block Comment

เป็น Comment หลายบรรทัด ที่ใช้ได้หลายบรรทัด การเขียน Comment ประเภทนี้ใช้เครื่องหมาย /* และ */ ข้อความใดที่อยู่ในเครื่องหมาย /* และ */ จะเป็น Comment ทั้งหมด เช่น

```
/* ต่อไปนี้เป็น Comment แบบหลายบรรทัด
บรรทัดนี้ก็เป็น Comment
Comment จะไม่มีผลต่อโปรแกรม
ภายใน Comment สามารถเขียนผิดหลักไวยากรณ์ภาษาได้
เพราะ Compilor ไม่สนใจ Comment
*/
```

```c
#include <stdio.h>

int main(void)
{

    /* This is
    block
    comment */
    printf("Hello world\n");

    return 0;
}
```

กฎการตั้งชื่อ

กฎการตั้งชื่อ

ภาษาซีมีกฎในการตั้งชื่อให้กับ Identifier ซึ่งได้แก่ ตัวแปร, ฟังก์ชัน และเลเบล ดังนี้

ชื่อที่ตั้งจะต้องไม่ซ้ำกับคำสงวน (Reserved Word)

asm	default	for	pascal	switch	_ds
auto	do	goto	register	typedef	_es
break	double	huge	return	union	_ss
case	else	if	short	unsigned	
cdecl	enum	int	signed	void	
char	extern	nterrupt	sizeof	volatile	
const	far	ong	static	while	
continue	float	near	struct	_cs	

กฎการตั้งชื่อ

- ชื่อต่าง ๆ ที่ตั้งจะเป็นแบบ Case-Sensitive หมายความว่าตัวอักษรใหญ่กับตัวอักษรเล็กถือว่าเป็นคนละตัวกัน เช่น TEST, Test, test, tEst ถือว่าเป็นคนละชื่อกัน

- ชื่อจะต้องขึ้นต้นด้วยตัวอักษรหรือเครื่องหมาย Underscore (_) เท่านั้น จะขึ้นต้นด้วยตัวเลขไม่ได้ แต่ภายในชื่อสามารถประกอบด้วยตัวอักษร เครื่องหมาย Underscore หรือตัวเลขก็ได้ เช่น TEST_VALUE, HELLO123, h1_T2, _UserName เป็นต้น

การตั้งชื่อจะเว้นวรรค (มีช่องว่างหรือแท็บภายในชื่อ) ไม่ได้

การตั้งชื่อจะประกอบด้วยอักขระพิเศษ เช่น $, @, #, &, ไม่ได้