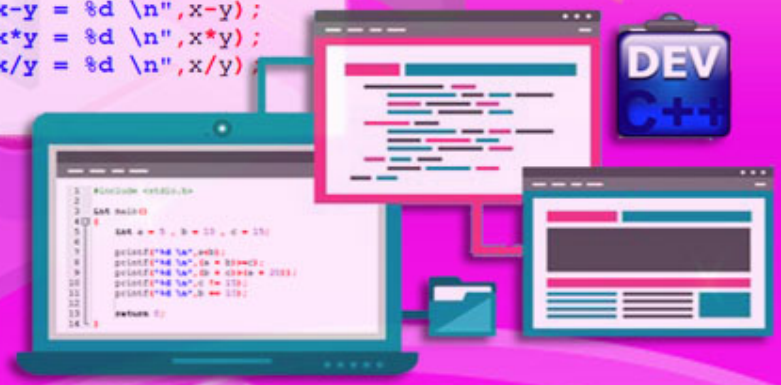
หลักสูตรการเขียนโปรแกรม

Coding Language

ภาษาซี (C)

```c
#include <stdio.h>

int main()
{
    int x=10;
    int y=5;
    printf("Hello World\n");
    printf("Variable Value = %d \n",x);
    printf("\t Value x+y = %d \n",x+y);
    printf("\t Value x-y = %d \n",x-y);
    printf("\t Value x*y = %d \n",x*y);
    printf("\t Value x/y = %d \n",x/y);
    return 0;
}
```

PROGRAMMING LANGUAGE



บริษัท 168 เอ็ดดูเคชั่น จำกัด

เลขที่ 128 อาคารพญาไทพลาซ่า ชั้นที่ 11 ห้องเลขที่ 117

ถนนพญาไท แขวงทุ่งพญาไท เขตราชเทวี กทม. โทร. 02-129-3208

คำนำ

จากนโยบายการศึกษาของกระทรวงศึกษาธิการที่มุ่งเน้นให้นักเรียนมีการเรียนรู้ในภาษาที่ 3 หรือภาษาโค้ดดิ้ง (Coding) โดยเริ่มตั้งแต่ชั้นประถมศึกษาปีที่ 1 เป็นต้นไป เพื่อส่งเสริมให้เกิด I-Innovation หรือ นวัตกรรมที่จะสามารถนำไปใช้ได้จริงและเกิดประโยชน์แก่คนหมู่มาก ทำให้ประเทศไทยมีบุคลากรที่สร้างนวัตกรรมหน้าใหม่ที่สามารถผลิตนวัตกรรมที่จะเปลี่ยนโลกได้ในทุกสาขาอาชีพ ซึ่งการพัฒนารูปแบบต่างๆ เหล่านี้จะต้องเป็นบุคคลที่มีความคิดสร้างสรรค์ มีความคิดที่เป็นระบบและเข้าใจในภาษาดิจิตอล เพื่อจะสร้างผู้นำทางด้านเทคโนโลยีใหม่ๆ การเรียนรู้โค้ดดิ้ง (Coding) จึงจำเป็นและสำคัญมากต่อการศึกษาชาติและโรงเรียนต่างๆ จำเป็นต้องพัฒนาการเรียนการสอนโค้ดดิ้งในทุกระดับชั้น

ภาษาซี (C Programming Language) คือ ภาษาคอมพิวเตอร์ใช้สำหรับพัฒนาโปรแกรมทั่วไป ถูกพัฒนาครั้งแรกเพื่อใช้เป็นภาษาสำหรับพัฒนาระบบปฏิบัติการยูนิกซ์ (Unix Opearating System) แทนภาษาแอสเซมบลี ซึ่งเป็นภาษาระดับต่ำที่สามารถกระทำในระบบฮาร์ดแวร์ได้ด้วยความรวดเร็ว ซึ่งภาษาซีเป็นที่นิยมในการเรียนรู้ในเรื่อง Coding เป็นอย่างมากเนื่องจากเป็นพื้นฐานของการเรียนรู้ในขั้นตอนของการเรียนการสอนการเขียนโปรแกรมในขั้นตอนต่อไป รวมทั้งสามารถใช้ภาษาซีร่วมกับการใช้งานของบอร์ดและอุปกรณ์ต่าง ๆ เพื่อทดสอบโปรแกรมและพัฒนาโปรแกรมในขั้นต่อไป

ซึ่งทางผู้จัดทำหวังเป็นอย่างยิ่งว่าสถานศึกษาที่ได้นำหลักสูตรการเรียนการสอน Coding เรื่อง หลักสูตร ภาษาซี นี้ไปเรียนรู้จะสามารถทำการเรียนการสอนหลักสูตร Coding ได้เป็นอย่างดีและสามารถประยุกต์ใช้กับอุปกรณ์อื่นๆได้ต่อไป

บริษัท 168 เอ็ดดูเคชั่น จำกัด

สารบัญ

คำอธิบายรายวิชา

วิชา การเขียนโปรแกรมภาษาซี กลุ่มสาระการเรียนรู้วิทยาศาสตร์

ระดับชั้นมัธยมศึกษา ภาคเรียนที่ ๑-๒ เวลา ๔๐ ชั่วโมง จำนวน ๑ หน่วยกิต

เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรมตัวชี้วัด

การพัฒนารูปแบบต่างๆ เหล่านี้จะต้องเป็นบุคคลที่มีความคิดสร้างสรรค์ มีความคิดที่เป็นระบบ และเข้าใจในภาษาดิจิตอล เพื่อจะสร้างผู้นำทางด้านเทคโนโลยีใหม่ๆ การเรียนรู้โค้ดดิ้ง (Coding) จึงจำเป็นและสำคัญมากต่อการศึกษาชาติและโรงเรียนต่างๆ จำเป็นต้องพัฒนาการเรียนการสอนโค้ดดิ้งในทุกระดับชั้น ภาษาซี (C Programming Language) คือ ภาษาคอมพิวเตอร์ใช้สำหรับพัฒนาโปรแกรมทั่วไป ถูกพัฒนาครั้งแรกเพื่อใช้เป็นภาษาสำหรับพัฒนาระบบปฏิบัติการยูนิกซ์ ที่สามารถกระทำในระบบฮาร์ดแวร์ได้ด้วยความรวดเร็ว มีความรู้ความเข้าใจในหลักการของการเขียนโปรแกรมด้วยภาษาซี เข้าใจคำสั่งและสามารถเขียนโปรแกรมเบื้องต้น รวมทั้งเชื่อมโยงและออกแบบโปรแกรมกับ Output ในรูปแบบต่าง ๆ ได้เป็นอย่างดี

ตัวชี้วัด

ว ๔.๑ ม.๑/๑ ม.๑/๒ ม.๑/๓ ม.๑/๔ ม.๒/๓ ม.๓/๓ ม.๔/๑ ม.๔/๓

ว ๔.๒ ม.๑/๑ ม.๑/๒ ม.๑/๓ ม.๒/๑ ม.๒/๒ ม.๒/๓ ม.๓/๑ ม.๓/๒ ม.๓/๓

รวม ๑๗ ตัวชี้วัด

ตัวชี้วัดและสาระการเรียนรู้

วิชา การเขียนโปรแกรมภาษาซี กลุ่มสาระการเรียนรู้วิทยาศาสตร์

ระดับชั้นมัธยมศึกษา ภาคเรียนที่ ๑-๒ เวลา ๔๐ ชั่วโมง จำนวน ๑ หน่วยกิต

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๑ เข้าใจแนวคิดหลักของเทคโนโลยีเพื่อการดำรงชีวิตในสังคมที่มีการเปลี่ยนแปลงอย่างรวดเร็ว ใช้ความรู้และทักษะทางด้านวิทยาศาสตร์ คณิตศาสตร์ และศาสตร์อื่น ๆ เพื่อแก้ปัญหาหรือพัฒนางานอย่างมีความคิดสร้างสรรค์ด้วยกระบวนการออกแบบเชิงวิศวกรรม เลือกใช้เทคโนโลยีอย่างเหมาะสม โดยคำนึงถึงผลกระทบต่อชีวิต สังคม และสิ่งแวดล้อม

ลำดับ	ตัวชี้วัด	สาระการเรียนรู้แกนกลาง
๑	ว ๔.๑ ม.๑/๑	อธิบายแนวคิดหลักของเทคโนโลยีในชีวิตประจำวันและวิเคราะห์สาเหตุหรือปัจจัยที่ส่งผลต่อการเปลี่ยนแปลงของเทคโนโลยี
๒	ว ๔.๑ ม.๑/๒	ระบุปัญหาหรือความต้องการในชีวิตประจำวัน รวบรวม วิเคราะห์ข้อมูลและแนวคิดที่เกี่ยวข้องกับปัญหา
๓	ว ๔.๑ ม.๑/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็น นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนและดำเนินการแก้ปัญหา
๔	ว ๔.๑ ม.๑/๔	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็น นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนและดำเนินการแก้ปัญหา
๖	ว ๔.๑ ม.๒/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็นภายใต้เงื่อนไขและทรัพยากรที่มีอยู่ นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนขั้นตอนการทำงานและดำเนินการแก้ปัญหาอย่างเป็นขั้นตอน
๘	ว ๔.๑ ม.๓/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็นภายใต้เงื่อนไขและทรัพยากรที่มีอยู่ นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจด้วยเทคนิคหรือวิธีการที่หลากหลาย วางแผนขั้นตอนการทำงานและดำเนินการแก้ปัญหาอย่างเป็นขั้นตอน

๑๑	ว ๔.๑ ม.๔/๑	วิเคราะห์แนวคิดหลักของเทคโนโลยี ความสัมพันธ์กับศาสตร์อื่น โดยเฉพาะวิทยาศาสตร์ หรือคณิตศาสตร์ รวมทั้งประเมินผลกระทบที่จะเกิดขึ้นต่อมนุษย์ สังคม เศรษฐกิจ และสิ่งแวดล้อม เพื่อเป็นแนวทางในการพัฒนาเทคโนโลยี
๑๒	ว ๔.๑ ม.๔/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็นภายใต้เงื่อนไขและทรัพยากรที่มีอยู่ นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจด้วยเทคนิคหรือวิธีการที่หลากหลาย โดยใช้ซอฟต์แวร์ช่วยในการออกแบบ วางแผนขั้นตอนการทำงานและดำเนินการแก้ปัญหา

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

ลำดับ	ตัวชี้วัด	สาระการเรียนรู้แกนกลาง
๑	ว ๔.๒ ม.๑/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงนามธรรมเพื่อแก้ปัญหาหรืออธิบายการทำงานที่พบในชีวิตจริง
๒	ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์
๓	ว ๔.๒ ม.๑/๓	รวบรวมข้อมูลปฐมภูมิ ประมวลผล ประเมินผล นำเสนอข้อมูล และสารสนเทศ ตามวัตถุประสงค์โดยใช้ซอฟต์แวร์ หรือบริการบนอินเทอร์เน็ต ที่หลากหลาย
๔	ว ๔.๒ ม.๑/๔	ใช้เทคโนโลยีสารสนเทศอย่างปลอดภัย ใช้สื่อและแหล่งข้อมูลตามข้อกำหนดและข้อตกลง
๕	ว ๔.๒ ม.๒/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงคำนวณในการแก้ปัญหา หรือการทำงานที่พบในชีวิตจริง
๖	ว ๔.๒ ม.๒/๒	ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา
๗	ว ๔.๒ ม.๓/๑	พัฒนาแอปพลิเคชันที่มีการบูรณาการกับวิชาอื่นอย่างสร้างสรรค์
๗	ว ๔.๒ ม.๓/๒	รวบรวมข้อมูล ประมวลผล ประเมินผล นำเสนอข้อมูลและสารสนเทศตามวัตถุประสงค์โดยใช้ซอฟต์แวร์หรือบริการบนอินเทอร์เน็ตที่หลากหลาย

๗	ว ๔.๒ ม.๓/๓	ประเมินความน่าเชื่อถือของข้อมูล วิเคราะห์สื่อและผลกระทบจากการให้ข่าวสารที่ผิดเพื่อการใช้งานอย่างรู้เท่าทัน

โครงสร้างรายวิชา

วิชา การเขียนโปรแกรมภาษาซี กลุ่มสาระการเรียนรู้วิทยาศาสตร์

ชั้นมัธยมศึกษา ภาคเรียนที่ ๑-๒ เวลา ๔๐ ชั่วโมง จำนวน ๑ หน่วยกิต

ลำดับ	ชื่อหน่วยการเรียนรู้	สาระสำคัญ (key Concept)	มาตรฐานการเรียนรู้/ ตัวชี้วัด	เวลา (ชม.)	น้ำหนักคะแนน
๑	ความรู้เบื้องต้นเกี่ยวกับภาษาซี	รู้จักกับโปรแกรมภาษาซี ประวัติและความเป็นมาในการสร้างและเริ่มใช้งาน	ว.๔.๑ ม.๑/๑-๔ ว ๔.๒ ม.๑/๒ ว ๔.๒ ม.๓/๑-๓ ว ๔.๒ ม.๔/๓	๓	๕
๒	โครงสร้างภาษาซี	เรียนรู้โครงสร้างของภาษาซี ส่วนประกอบและใช้งานเบื้องต้น	ว ๔.๑ ม.๑/๓ ว ๔.๑ ม.๓/๓ ว ๔.๑ ม.๔/๓ ว ๔.๒ ม.๑/๒	๓	๕
๓	ตัวแปร ชนิดข้อมูล และตัวดำเนินการ	เรียนรู้เกี่ยวกับตัวแปร ชนิดข้อมูลและตัวดำเนินการในภาษาซี	ว ๔.๑ ม.๔/๑ ว ๔.๒ ม.๑/๒ ว ๔.๒ ม.๒/๒	๔	๕
๔	การรับและแสดงผลข้อมูล	เรียนรู้วิธีกำหนดและแสดงการรับและแสดงผลข้อมูลในโปรแกรม	ว ๔.๒ ม.๑/๑-๓ ว ๔.๒ ม.๒/๑-๒	๔	๕
๕	ขั้นตอนการทำงานของโปรแกรม	เรียนรู้และปฏิบัติเกี่ยวกับขั้นตอนการทำงานของโปรแกรมอย่างเป็นขั้นตอน	ว ๔.๑ ม.๑/๓-๔ ว ๔.๒ ม.๑/๑-๓ ว ๔.๒ ม.๒/๑-๒	๔	๕
๖	คำสั่งควบคุม	เรียนรู้และฝึกปฏิบัติเกี่ยวกับคำสั่งควบคุมของภาษาซี	ว ๔.๑ ม.๑/๓ ว ๔.๑ ม.๒/๓ ว ๔.๒ ม.๑/๑-๓ ว ๔.๒ ม.๒/๒	๔	๕

ลำดับ	ชื่อหน่วยการเรียนรู้	สาระสำคัญ (key Concept)	มาตรฐานการเรียนรู้/ ตัวชี้วัด	เวลา (ชม.)	น้ำหนักคะแนน
๗	พอยน์เตอร์และอาร์เรย์	รู้จักกับพอยน์เตอร์และอาร์เรย์และการใช้งานร่วมกัน	ว ๔.๒ ม.๑/๒ ว ๔.๒ ม.๒/๒	๓	๕
๘	ฟังก์ชั่นและไลบรารี่	สามารถใช้ฟังก์ชั่นและไลบรารี่ในส่วนของคำสั่งได้	ว ๔.๒ ม.๑/๒ ว ๔.๒ ม.๒/๑ ว ๔.๒ ม.๒/๒	๔	๕
๙	สตริงและสตรักเจอร์	เรียนรู้เรื่องสตริงและสตรักเจอร์	ว ๔.๒ ม.๑/๒ ว ๔.๒ ม.๒/๑ ว ๔.๒ ม.๒/๒	๔	๕
๑๐	การจัดการแฟ้มข้อมูลในภาษาซี	สรุปใจความสำคัญและวิธีใช้งานของหลัก	ว ๔.๒ ม.๑/๒ ว ๔.๒ ม.๒/๒	๔	๕
กิจกรรม				๑	๑๐
ข้อสอบกลางภาค				๑	๒๐
ข้อสอบปลายภาค				๑	๒๐
รวม				๔๐	๑๐๐

หน่วยการเรียนรู้ที่ ๑ ความรู้เบื้องต้นเกี่ยวกับภาษาซี

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๑ เข้าใจแนวคิดหลักของเทคโนโลยีเพื่อการดำรงชีวิตในสังคมที่มีการเปลี่ยนแปลงอย่างรวดเร็ว ใช้ความรู้และทักษะทางด้านวิทยาศาสตร์ คณิตศาสตร์ และศาสตร์อื่น ๆ เพื่อแก้ปัญหาหรือพัฒนางานอย่างมีความคิดสร้างสรรค์ด้วยกระบวนการออกแบบเชิงวิศวกรรม เลือกใช้เทคโนโลยีอย่างเหมาะสมโดยคำนึงถึงผลกระทบต่อชีวิต สังคม และสิ่งแวดล้อม

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๑ ม.๑/๑	อธิบายแนวคิดหลักของเทคโนโลยีในชีวิตประจำวันและวิเคราะห์สาเหตุหรือปัจจัยที่ส่งผลต่อการเปลี่ยนแปลงของเทคโนโลยี
ว ๔.๑ ม.๑/๒	ระบุปัญหาหรือความต้องการในชีวิตประจำวัน รวบรวม วิเคราะห์ข้อมูลและแนวคิดที่เกี่ยวข้องกับปัญหา
ว ๔.๑ ม.๑/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็น นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนและดำเนินการแก้ปัญหา
ว ๔.๒ ม.๓/๑	พัฒนาแอปพลิเคชันที่มีการบูรณาการกับวิชาอื่นอย่างสร้างสรรค์
ว ๔.๒ ม.๓/๒	รวบรวมข้อมูล ประมวลผล ประเมินผล นำเสนอข้อมูลและสารสนเทศตามวัตถุประสงค์โดยใช้ซอฟต์แวร์หรือบริการบนอินเทอร์เน็ตที่หลากหลาย
ว ๔.๒ ม.๓/๓	ประเมินความน่าเชื่อถือของข้อมูล วิเคราะห์สื่อและผลกระทบจากการให้ข่าวสารที่ผิด เพื่อการใช้งานอย่างรู้เท่าทัน
ว ๔.๒ ม.๓/๔	ใช้เทคโนโลยีสารสนเทศอย่างปลอดภัย และมีความรับผิดชอบต่อสังคม ปฏิบัติตามกฎหมายเกี่ยวกับคอมพิวเตอร์ ใช้ลิขสิทธิ์ของผู้อื่นโดยชอบธรรม

ว ๔.๒ ม.๔/๓ ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็นภายใต้เงื่อนไขและทรัพยากรที่มีอยู่ นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจด้วยเทคนิคหรือวิธีการที่หลากหลาย โดยใช้ซอฟต์แวร์ช่วยในการออกแบบ วางแผนขั้นตอนการทำงานและดำเนินการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความรู้จักกับความเป็นมาและวิวัฒนาการของภาษาซี

๒. เข้าใจหลักการทำงานและวิเคราะห์ความแตกต่างของการทำงานในภาษาซี

๓. ทำความรู้จักกับโปรแกรมคอมไพเลอร์ในภาษาซี

๔. ทำความรู้จักและเริ่มต้นใช้งานโปรแกรมไอดีอี

<u>สาระการเรียนรู้</u>

- **ความรู้**

๑. ความรู้เกี่ยวกับความเป็นมาของภาษาซี

๒. ความรู้เกี่ยวกับหลักการแปลภาษาของคำสั่งในภาษาซี

๓. รู้จักและทดลองการใช้โปรแกรมไอดีอีได้แก่ โปรแกรม DEV C++

- **ทักษะ / กระบวนการ**

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และ จำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 1 ทำความรู้จักกับภาษาซี

จุดเริ่มต้นของภาษาซี

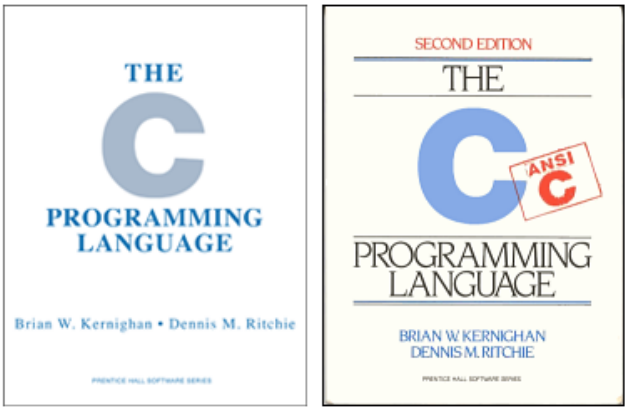
ภาษาซี หรือ C Language คือ เป็นภาษาคอมพิวเตอร์ที่ใช้สำหรับพัฒนาโปรแกรมทั่วไป ภาษาซีเกิดขึ้นในปี ค.ศ.1972 ได้รับการออกแบบและพัฒนาขึ้นโดย เดนนิส ริดชี (Dennis Ritchie) แห่งห้องทดลองเบลล์ (Bell Laboratories) ที่เมอร์รีฮิล มลรัฐนิวเจอร์ซี่ ประเทศสหรัฐอเมริกา ภาษาซีนั้นพัฒนามาจาก “ภาษา B” ซึ่งพัฒนาขึ้นโดยเคน ทอมสัน (Ken Tomson) ภาษา B ถูกพัฒนาบนพื้นฐานของภาษาบีซีพีแอล (BCPL) พัฒนาขึ้นโดย มาร์ติน ริชาร์ด (Martin Richards) การออกแบบและพัฒนาภาษาซีของเดนนิส ริดชี มีจุดมุ่งหมายให้เป็นภาษาสำหรับใช้เขียนโปรแกรมปฏิบัติการระบบยูนิกซ์ และได้ตั้งชื่อว่า ซี (C) เพราะเห็นว่า ซี (C) เป็นตัวอักษรต่อจากบี (B)

Dennis Ritchie ผู้ออกแบบและพัฒนาภาษาซี (C)

ที่มารูปภาพ: http://kruviewly.blogspot.com/p/blog-page_8169.html

วิวัฒนาการของภาษาซี

ภาษาC เริ่มมีคนสนใจมากขึ้นในปี ค.ศ.1978 เมื่อ Brain Kernighan ร่วมกับ Dennis Ritchie พัฒนามาตรฐานของภาษาซีขึ้นมา คือ K&R (Kernighan & Ritchie) และทั้งสองยังได้แต่งหนังสือชื่อว่า “The C Programming Language” หนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ในการเขียนโปรแกรมมากขึ้น ในเวลาต่อมาภาษาซีได้รับความนิยมสูง สถาบัน ANSI (American National Standards Institute) ได้สร้างมาตรฐานภาษาซีขึ้นมา เพื่อรับรองให้เป็นสากล ภายใต้ชื่อว่า ANSI-C ตั้งแต่ปี ค.ศ.1983



หนังสือ “The C Programming Language”

ที่มาภาพ : https://en.wikipedia.org/wiki/The_C_Programming_Language

ในปัจจุบันได้มีการพัฒนาภาษาซีให้มีประสิทธิภาพมากยิ่งขึ้น เป็นเวอร์ชันต่าง ๆ มากมาย มีการพัฒนาต่อยอดเป็นภาษาซีพลัสพลัส (C++) หรือภาษาซีชาร์ป (C#) ซึ่งมีการเพิ่มชุดคำสั่งที่สนับสนุนการพัฒนาโปรแกรมเชิงวัตถุ (Object-Oriented Programming) และยังคงรองรับชุดคำสั่งมาตรฐานของภาษาซี คือ ANSI-C อยู่ด้วย

ภาษาซีเป็นภาษาแบบโครงสร้างที่สามารถศึกษาและทำความเข้าใจได้ไม่ยาก อีกทั้งยังสามารถเป็นพื้นฐานในการเขียนโปรแกรมภาษาอื่นๆ ได้อีก เช่น C++, Perl, JAVA, PHP, Python, Ruby เป็นต้น ซึ่งส่วนใหญ่ได้รับความนิยมนำมาใช้เขียนโปรแกรม จึงเป็นเหตุผลหนึ่งที่ทำให้ภาษาซีเป็นภาษาแรกในการเริ่มต้นโดยเฉพาะการทำงานที่มีข้อจำกัดในเรื่องหน่วยความจำ และการประมวลผล

วิวัฒนาการของภาษาซี

- ค.ศ. 1970 มีการพัฒนาภาษา B โดย Ken Thompson ซึ่งทำงานบนเครื่อง DEC PDP-7 ซึ่ง ทำงานบนเครื่องไมโครคอมพิวเตอร์ไม่ได้ และยังมีข้อจำกัดในการใช้งานอยู่ (ภาษา B สืบทอด มาจาก ภาษา BCPL ซึ่งเขียนโดย Marth Richards)

- ค.ศ. 1972 Dennis M. Ritchie และ Ken Thompson ได้สร้างภาษา C เพื่อเพิ่มประสิทธิภาพ ภาษา B ให้ดียิ่งขึ้น ในระยะแรกภาษา C ไม่เป็นที่นิยมแก่นักโปรแกรมเมอร์โดยทั่วไปนัก

- ค.ศ. 1978 Brian W. Kernighan และ Dennis M. Ritchie ได้เขียนหนังสือเล่มหนึ่งชื่อว่า The C Programming Language และหนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ในการเขียนโปรแกรมมากขึ้น

- แต่เดิมภาษา C ใช้ Run บนเครื่องคอมพิวเตอร์ 8 bit ภายใต้ระบบปฏิบัติการ CP/M ของ IBM PC ซึ่งในช่วงปี ค. ศ. 1981 เป็นช่วงของการพัฒนาเครื่องไมโครคอมพิวเตอร์ ภาษา C จึงมี บทบาทสำคัญในการนำมาใช้บนเครื่อง PC ตั้งแต่นั้นเป็นต้นมา และมีการพัฒนาต่อมาอีกหลายๆ ค่าย ดังนั้นเพื่อกำหนดทิศทางการใช้ภาษา C ให้เป็นไปแนวทางเดียวกัน ANSI (American National Standard Institute) ได้กำหนดข้อตกลงที่เรียกว่า 3J11 เพื่อสร้างภาษา C มาตรฐานขึ้นมา เรียกว่า ANSI C

- ค.ศ. 1983 Bjarne Stroustrup แห่งห้องปฏิบัติการเบล (Bell Laboratories) ได้พัฒนาภาษา C++ ขึ้น รายละเอียดและความสามารถของ C++ มีส่วนขยายเพิ่มจาก C ที่สำคัญ ๆ ได้แก่ แนวความคิดของการเขียนโปรแกรมแบบกำหนดวัตถุเป้าหมายหรือแบบ OOP (Object Oriented Programming) ซึ่งเป็นแนวการเขียนโปรแกรมที่เหมาะกับการพัฒนาโปรแกรมขนาดใหญ่ที่มี ความสลับซับซ้อนมาก มีข้อมูลที่ใช้ในโปรแกรมจานวนมาก จึงนิยมใช้เทคนิคของการเขียนโปรแกรมแบบ OOP ในการพัฒนาโปรแกรมขนาดใหญ่ในปัจจุบันนี้

ลักษณะเด่นของภาษาซี

1. สามารถใช้งานบนสภาพแวดล้อมที่แตกต่างกันได้ ซอร์สโค้ดภาษาซี สามารถรันอยู่บนคอมพิวเตอร์ได้หลากหลายระดับ ตั้งแต่ระดับเมนเฟรมไปจนถึงไมโครเฟรม โดยไม่ต้องเปลี่ยนแปลงชุดคำสั่งใดๆ หรือเปลี่ยนแปลงเพียงเล็กน้อย

2. มีประสิทธิภาพสูง ชุดคำสั่งมีความกะทัดรัดและกระชับมาก และยังมีความใกล้ชิดกับฮาร์ดแวร์มากกว่าภาษาอื่นๆ

3. ความสามารถในการโปรแกรมโมดูล ภาษาซีมีการแบ่งโมดูลเพื่อคอมไพล์ได้ และยังสามารถลิงก์เชื่อมโยงเข้าด้วยกันได้อีก ภาษาซีประกอบด้วยฟังก์ชัน โดยโมดูลต่างๆ จะเขียนอยู่ในรูปของฟังก์ชันทั้งสิ้น

4. มีความยืดหยุ่นสูง สามารถใช้งานร่วมกับภาษาอื่นๆได้ มีการกล่าวว่า "ภาษา C เป็นภาษาที่อยู่กึ่งกลางระหว่างภาษาระดับต่ำและภาษาระดับสูง"

5. ตัวอักษรตัวพิมพ์เล็กและตัวอักษรพิมพ์ใหญ่แตกต่างกัน ตามปกติภาษาระดับสูงทั่วไป ตัวแปรที่ตั้งขึ้นด้วยตัวอักษรพิมพ์เล็กและตัวพิมพ์ใหญ่ สามารถนำมาใช้ร่วมกันได้ แต่ในภาษาซีถือว่าแตกต่างกัน เช่น NUM ไม่เท่ากับ num

ตัวแปลภาษาคอมพิวเตอร์

ภาษาซีเป็นภาษาโปรแกรมที่ต้องใช้ตัวแปลภาษาในการ ช่วยแปลจากโปรแกรมต้นฉบับเป็นโปรแกรมจุดหมาย นั่นคือภาษาเครื่อง เพราะคอมพิวเตอร์รับรู้เฉพาะภาษาเครื่องได้เพียงภาษาเดียวเท่านั้น

ตัวแปลภาษามี 2 ประเภท คือ คอมไพเลอร์ (Compiler) และอินเทอร์พรีเตอร์ (Interpreter)

1. คอมไพเลอร์ (Compiler) ทำการตรวจสอบความถูกต้องของการเขียนคำสั่งทั้งโปรแกรมให้เป็นออบเจ็คโค้ด แล้วจึงทำการแปลคำสั่งไปเป็นภาษาเครื่อง จากนั้นจึงทำการประมวลผลและแสดงผลลัพธ์ หากพบข้อผิดพลาดของการเขียนโปรแกรม หรือมีคำสั่งที่ผิดหลักไวยากรณ์ของภาษาคอมพิวเตอร์ โปรแกรมคอมไพเลอร์จะแจ้งให้ผู้เขียนทำการแก้ไขให้ถูกต้องทั้งหมดก่อน แล้วจึงคอมไพล์ใหม่อีกครั้งจนกว่าไม่พบข้อผิดพลาดถึงจะนำโปรแกรมไปใช้งานได้

2. อินเทอร์พรีเตอร์ (Interpreter) การอ่านและแปลโปรแกรมทีละบรรทัด เมื่อแปลผลบรรทัดหนึ่งเสร็จก็จะทำงานตามคำสั่งในบรรทัดนั้น แล้วจึงทำการแปลผลตามคำสั่งในบรรทัดถัดไป เหมาะสำหรับโปรแกรมที่ไม่ยาวมาก และต้องการผลลัพธ์ทันที

- **ข้อดีและข้อเสียของตัวแปลภาษาทั้งสองแบบมีดังนี้**

ตัวแปลภาษา	ข้อดี	ข้อเสีย
คอมไพเลอร์	ทำงานได้เร็ว เนื่องจากทำการแปลผลทีเดียวแล้วจึงทำงานตามคำสั่งของโปรแกรมในภายหลัง	เมื่อเกิดข้อผิดพลาดขึ้นกับโปรแกรมจะตรวจสอบหาข้อผิดพลาดได้ยาก เพราะทำการแปลผลทีเดียวทั้งโปรแกรม
	เมื่อทำการแปลผลแล้ว ในครั้งต่อไปไม่จำเป็นต้องทำการแปลผลใหม่อีก เนื่องจากภาษาเครื่องที่แปลได้จะถูกเก็บไว้ที่หน่วยความจำ สามารถเรียกใช้งานได้ทันที	
อินเตอร์พรีเตอร์	หาข้อผิดพลาดของโปรแกรมได้ง่าย เนื่องจากทำการแปลผลทีละบรรทัด	ช้า เนื่องจากที่ทำงานทีละบรรทัด
	เนื่องจากทำงานทีละบรรทัดดังนั้นจึงสั่งให้โปรแกรมทำงานตามคำสั่งเฉพาะจุดที่ต้องการได้	
	ไม่เสียเวลารอการแปลโปรแกรมเป็นเวลานาน	

- **การทำงานของ Compiler จะประกอบด้วย 5 ขั้นตอนหลัก**

1. การวิเคราะห์คำศัพท์ (Lexical Analyzer) เป็นขั้นตอนในการตรวจจับและจัดเรียงคำของซอร์สโค้ดที่ต้องการแปลภาษา

2. การวิเคราะห์โครงสร้างภาษา (Syntax Analyzer) เป็นขั้นตอนในการตรวจสอบไวยากรณ์ของภาษาว่าถูกต้องตามรูปแบบของภาษานั้นๆ หรือไม่

3. การวิเคราะห์ความหมายของภาษา (Semantic Analyzer) เป็นขั้นตอนในการตรวจสอบชนิดของข้อมูลที่จะนำมา ประมวลผล และทำการเปลี่ยนรูปแบบของภาษาให้อยู่ในรูปของ Intermediate Form เพื่อรอการแปลงให้เป็น Object Program

4. การทำให้ตัวแทนภาษากลางมีความเหมาะสม (Code Generation) เป็นการแปล Intermediate Form ให้เป็น Object Program ซึ่ง ส่วนใหญ่จะเป็นภาษาแอสแซมบลี

5. การทำให้รหัสเป้าหมายมีความเหมาะสม (Code Optimization) เป็นการลดขั้นตอนการทำงานที่ซ้ำซ้อนกันของภาษาเครื่อง เพื่อให้การประมวลผลโดยเครื่องคอมพิวเตอร์ทำงานได้เร็วขึ้น

โปรแกรมไอดีอีภาษาซี

ในการเขียนโปรแกรม จะต้องใช้โปรแกรมอีดิเตอร์เขียนซอร์สโค้ด และแปลงไปเป็นภาษาเครื่อง ซึ่งต้องใช้โปรแกรมคอมไพเลอร์ ดังนั้นจึงนำไอดีอีมาใช้งาน เป็นเครื่องมือที่ประกอบด้วย 3 ส่วนหลัก คือ 1) ส่วนที่เป็นอีดิเตอร์เพื่อใช้เขียนซอร์สโค้ด 2) ส่วนที่เป็นคอมไพเลอร์เพื่อแปลงไปเป็นภาษาเครื่อง 3) ส่วนการทดสอบหาข้อผิดพลาดหรือการดีบัก (Debugging) ซึ่งภาษาซีมีโปรแกรมไอดีอีให้เลือกหลายตัวและเป็นไปตามมาตรฐาน ANSI C ซึ่งมีรายละเอียดเพิ่มเติมแตกต่างกันไป

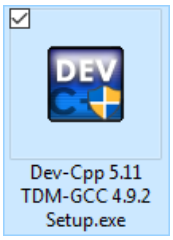
โปรแกรมที่สะดวก ง่ายในการคอมไพล์และสร้างรหัสเป็นภาษาเครื่อง สามารถใช้ Editor และ Compiler ที่รวมกันไว้แล้ว ในชุดพัฒนาหรือเครื่องมือที่ช่วยในการพัฒนาโปรแกรม เรียกว่า IDE (Integrated Development Environment) ซึ่งเป็นโปรแกรมที่ออกแบบมาเพื่อช่วยให้ผู้ที่เขียนโปรแกรมใช้ในการสร้างโปรแกรม โดยจะมี Editor สำหรับเขียนโค้ดของโปรแกรมและมีตัวแปลภาษามาพร้อม ปัจจุบันมีการออก

ชุดพัฒนำมาหลายรุ่นและเปลี่ยนแปลงอย่างรวดเร็ว เช่น Pelles C, Dev-C++, Turbo C++ , Borland C++ , Microsoft C/C++ , Microsoft Visual C++ , Microsoft Visual C# , Microsoft Visual C++ .NET ซึ่งชุดพัฒนาแต่ละตัวมีวิธีการนำไปใช้งานที่แตกต่างกัน เพราะพัฒนำมาจากบริษัทต่างกัน แต่อย่างไรก็ตามการเขียนโปรแกรมภาษาซีไม่ว่าจะเป็น IDE ใด ก็มีหลักและวิธีการที่คล้ายกัน จะต่างกันบ้างตรงรายละเอียดบางอย่างที่พัฒนาให้ใช้งานได้ง่ายขึ้น

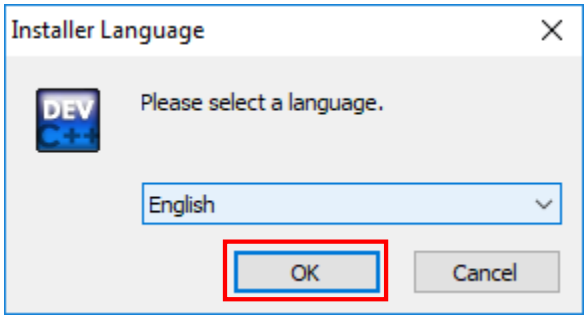
ในหลักสูตรนี้จะใช้ IDE คือ Bloodshed Dev-C++ ซึ่งสามารถเขียนได้ทั้งภาษา C และภาษา C++ เป็นชุดพัฒนาขึ้นมาเพื่อใช้เป็นฟรีแวร์ และทำงานภายใต้ระบบปฏิบัติการ Windows ทั้ง 32 บิต และ 64 บิต

การติดตั้งโปรแกรม

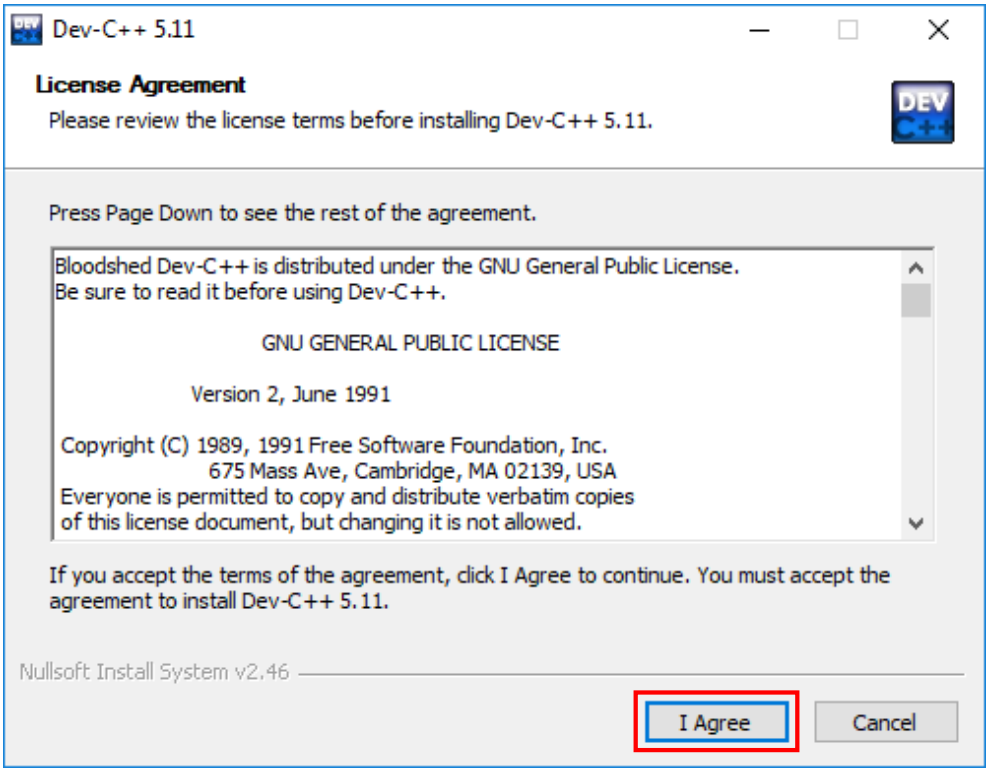
1. เปิดโฟลเดอร์ที่เก็บไฟล์โปรแกรม Dev-C++ ที่ดาวน์โหลดมา ดังภาพ



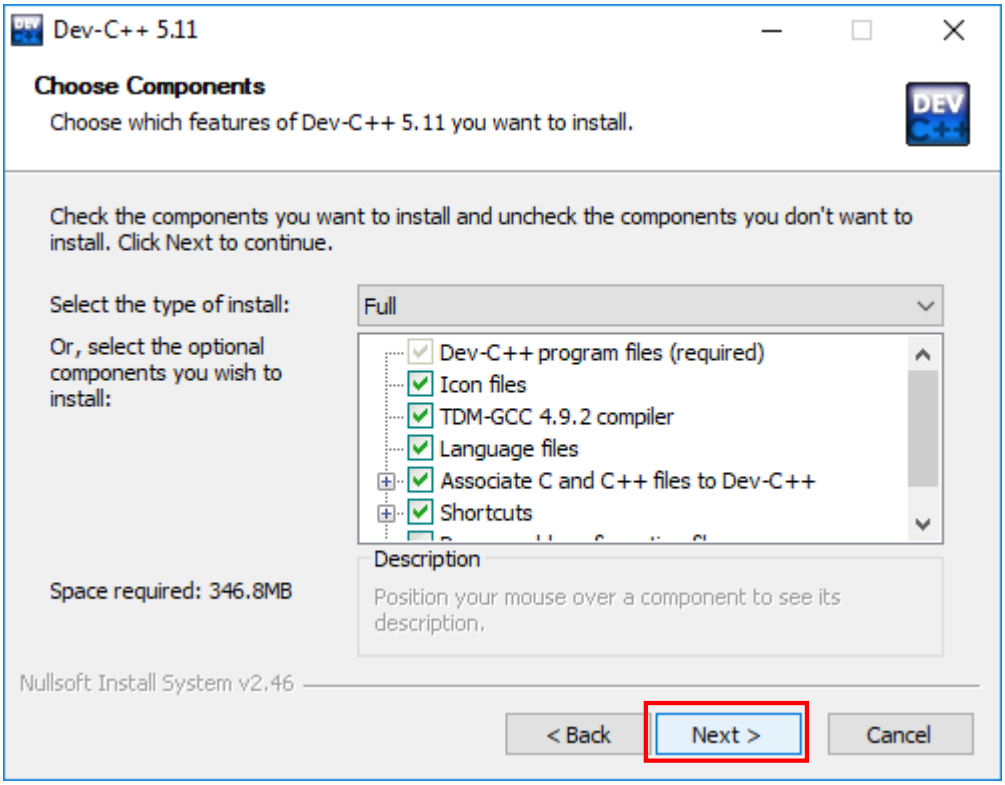
2. ดับเบิ้ลคลิกที่ไฟล์ เพื่อเริ่มทำการติดตั้งโปรแกรมลงเครื่อง จากนั้นจะเริ่มต้นขั้นตอนการติดตั้งโปรแกรม Dev-C++ โดยจะแสดงหน้าจอให้เลือกภาษา ให้เลือก English จากนั้นกดปุ่ม OK



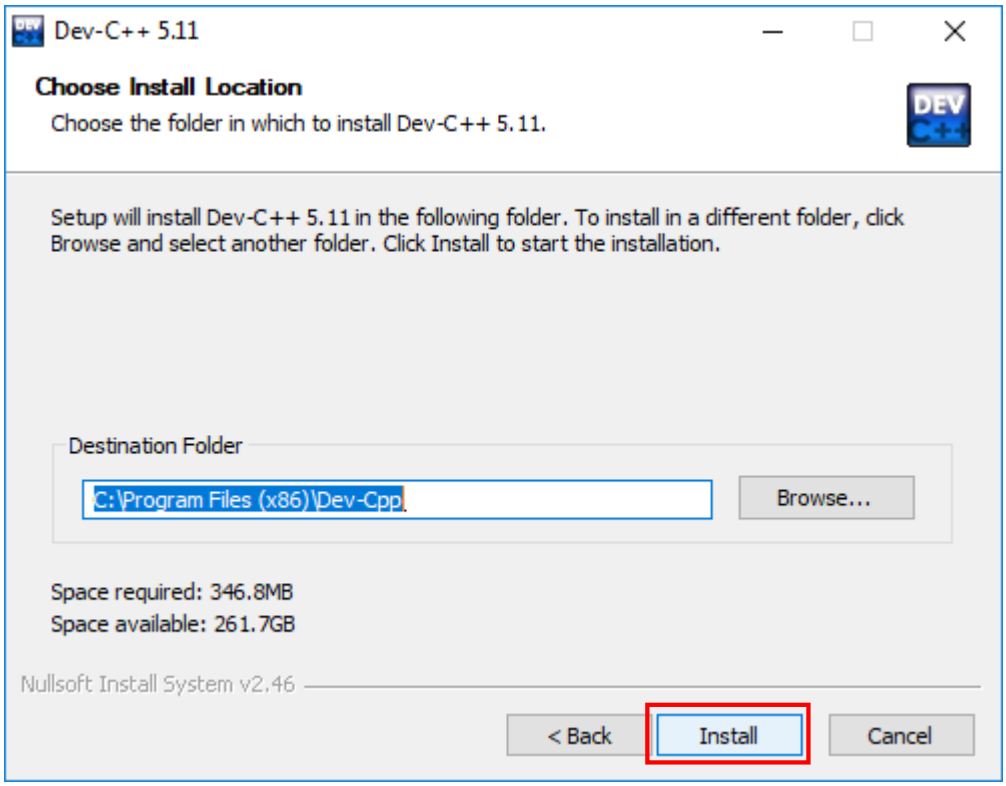
3. เมื่อเลือกภาษาเสร็จแล้ว จะแสดงหน้าจอ License Agreement (ข้อตกลงในการใช้โปรแกรม) ให้กดปุ่ม I Agree



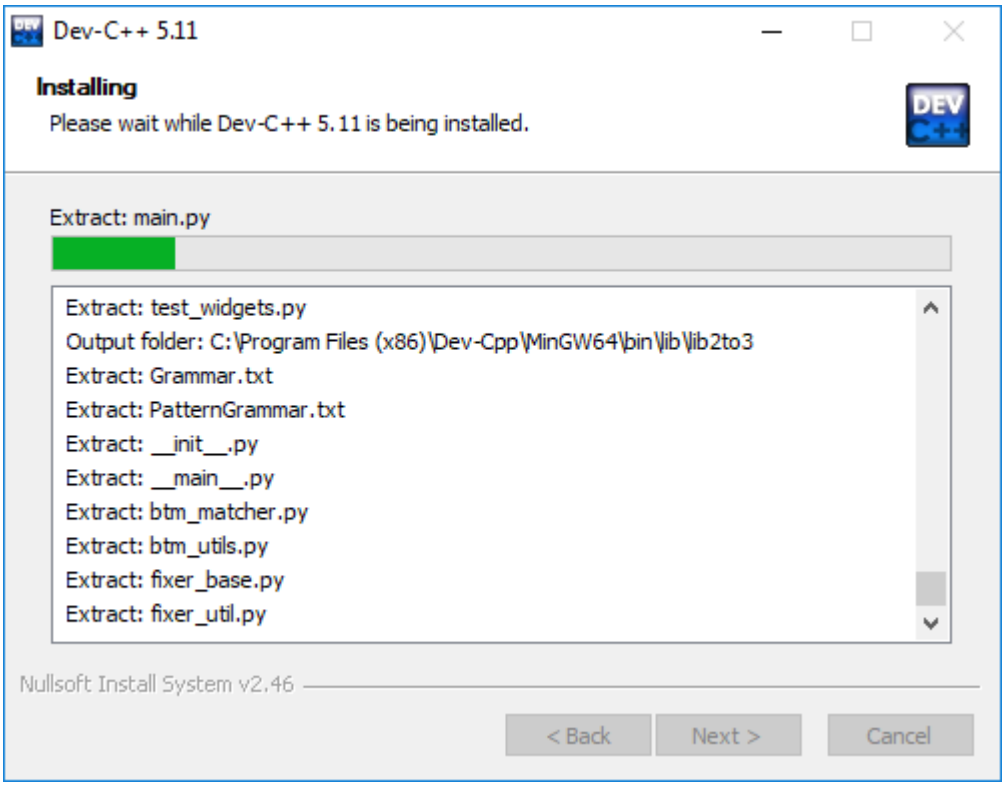
4. จากนั้นจะแสดงหน้าจอ Choose Components ให้กดปุ่ม Next >



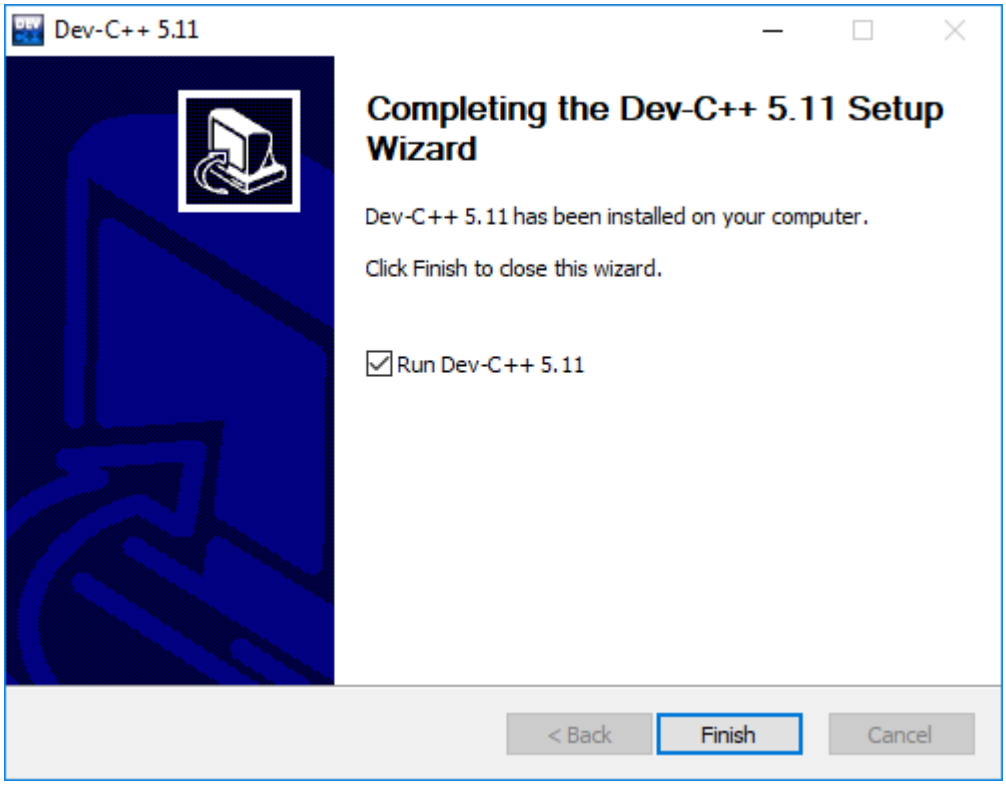
5. จากนั้นจะแสดงหน้าจอ Choose Install Location ให้กดปุ่ม Install ได้เลย



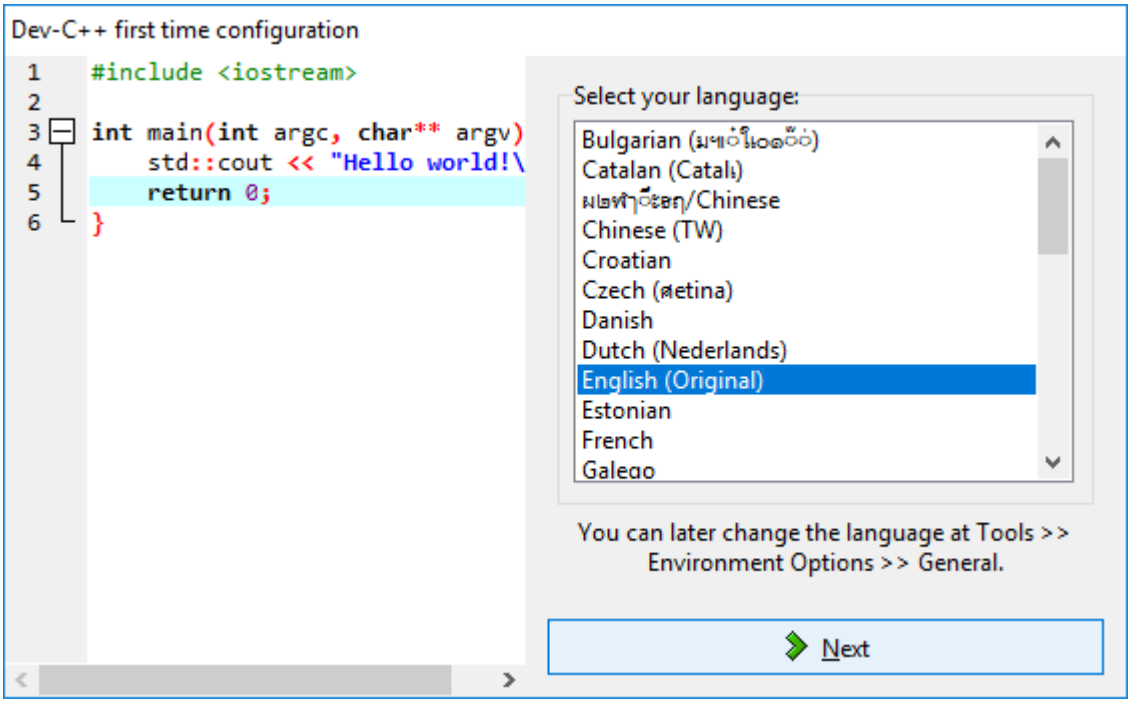
6. จากนั้นจะแสดงหน้าจอ installing รอสักครู่จนกว่าจะเสร็จ ซึ่งโปรแกรมกำลังถูกติดตั้งลงเครื่อง



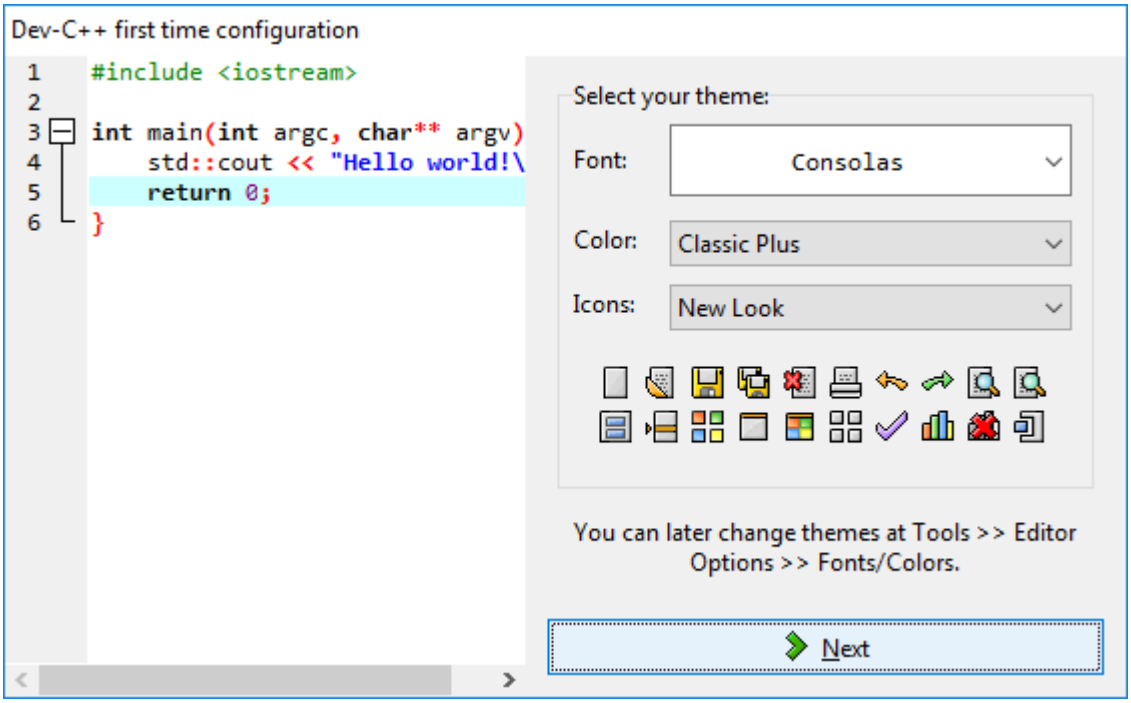
7. จากนั้นจะแสดงหน้าจอ completing the Dev-C++ 5.11 Setup Wizard เพื่อบอกว่าโปรแกรมได้ถูกติดตั้งเสร็จเรียบร้อยแล้ว ให้กดปุ่ม Finish



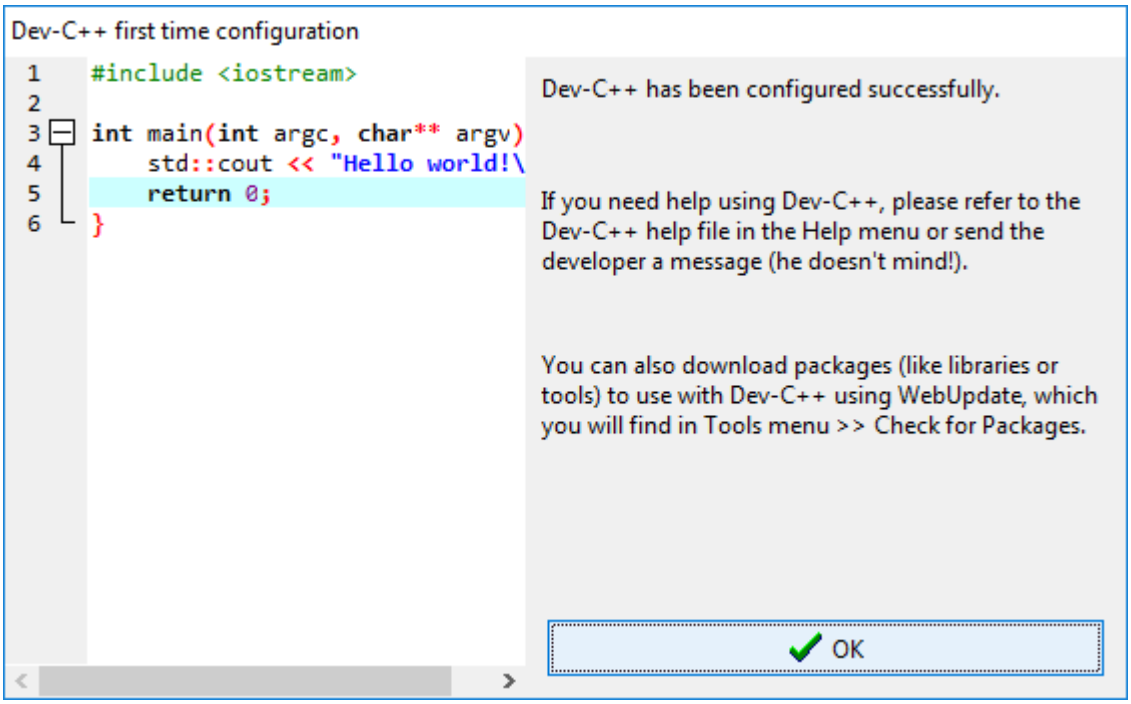
8. จากนั้นโปรแกรม Dev-C++ จะเปิดขึ้นมา เพื่อให้เรากำหนดค่าเริ่มต้นก่อนการใช้งาน เกี่ยวกับภาษาให้เลือก English (Original) แล้วกดปุ่ม Next



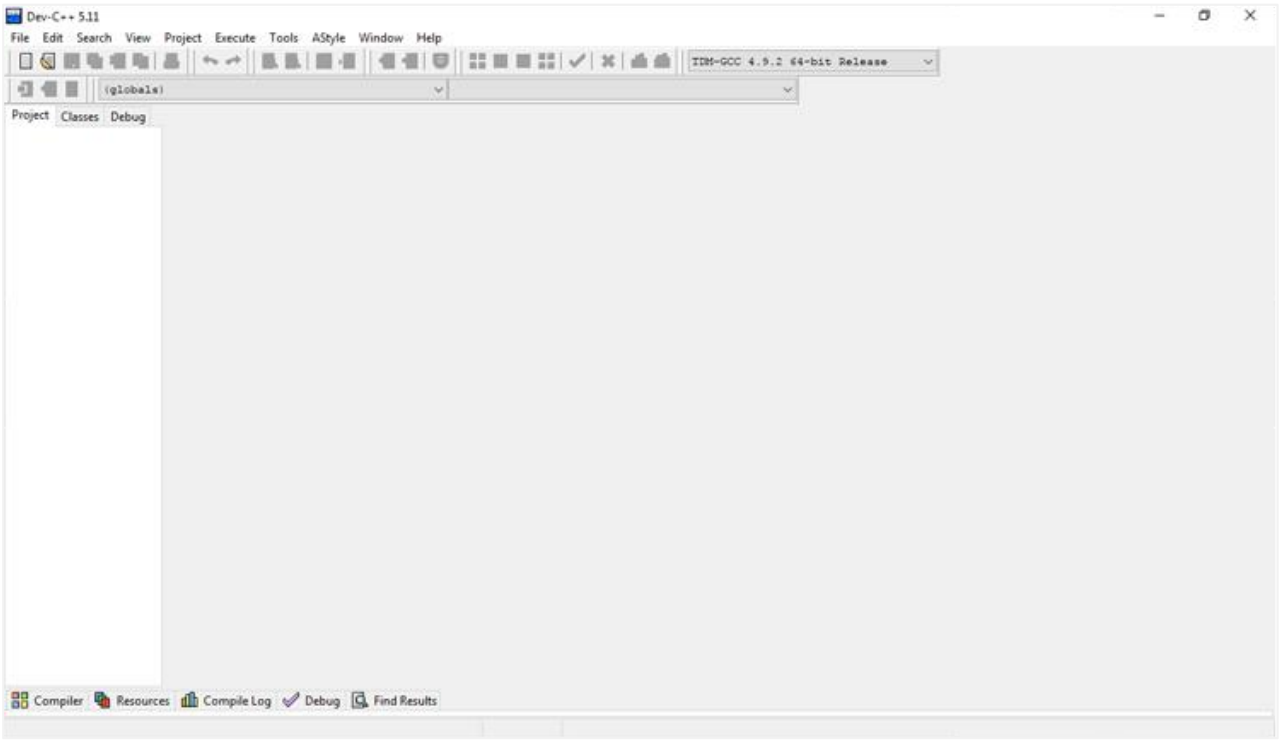
9. จากนั้นจะแสดงหน้าจอให้เลือกรูปแบบหน้าจอของโปรแกรมสามารถเลือกได้ตามใจชอบ จากนั้นให้กดปุ่ม Next



10. จากนั้นจะแสดงหน้าจอการตั้งค่าเสร็จสิ้นแล้ว ให้กดปุ่ม OK



11. จากนั้นจะแสดงหน้าจอของโปรแกรม Dev-C++ พร้อมให้เขียนโค้ดคำสั่งภาษาซีลงไป



12. เมื่อติดตั้งโปรแกรม Dev-C++ 5.11 ลงในเครื่องแล้ว จะปรากฏไอคอน 

สำหรับเข้าสู่โปรแกรมอยู่บน Desktop ของเครื่องคอมพิวเตอร์

หน่วยการเรียนรู้ที่ ๒ โครงสร้างภาษาซี

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๑ เข้าใจแนวคิดหลักของเทคโนโลยีเพื่อการดำรงชีวิตในสังคมที่มีการเปลี่ยนแปลงอย่างรวดเร็ว ใช้ความรู้และทักษะทางด้านวิทยาศาสตร์ คณิตศาสตร์ และศาสตร์อื่น ๆ เพื่อแก้ปัญหาหรือพัฒนางานอย่างมีความคิดสร้างสรรค์ด้วยกระบวนการออกแบบเชิงวิศวกรรม เลือกใช้เทคโนโลยีอย่างเหมาะสมโดยคำนึงถึงผลกระทบต่อชีวิต สังคม และสิ่งแวดล้อม

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๑ ม.๑/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็น นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนและดำเนินการแก้ปัญหา
ว ๔.๑ ม.๓/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็นภายใต้เงื่อนไขและทรัพยากรที่มีอยู่ นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจด้วยเทคนิคหรือวิธีการที่หลากหลาย วางแผนขั้นตอนการทำงานและดำเนินการแก้ปัญหาอย่างเป็นขั้นตอน
ว ๔.๑ ม.๔/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็นภายใต้เงื่อนไขและทรัพยากรที่มีอยู่ นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจด้วยเทคนิคหรือวิธีการที่หลากหลาย โดยใช้ซอฟต์แวร์ช่วยในการออกแบบ วางแผนขั้นตอนการทำงานและดำเนินการแก้ปัญหา
ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความเข้าใจลักษณะโครงสร้างภาษาซี

๒. เรียนรู้ส่วนประกอบของงภาษาซี

๓. เรียนรู้กฎเกณฑ์การเขียนภาษาซี

<u>สาระการเรียนรู้</u>

- **ความรู้**

๑. ความรู้เกี่ยวกับลักษณะโครงสร้างภาษาซี

๒. ความรู้เกี่ยวกับส่วนประกอบของงภาษาซี

๓. เข้าใจถึงกฎเกณฑ์การเขียนภาษาซี

- **ทักษะ / กระบวนการ**

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และ จำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

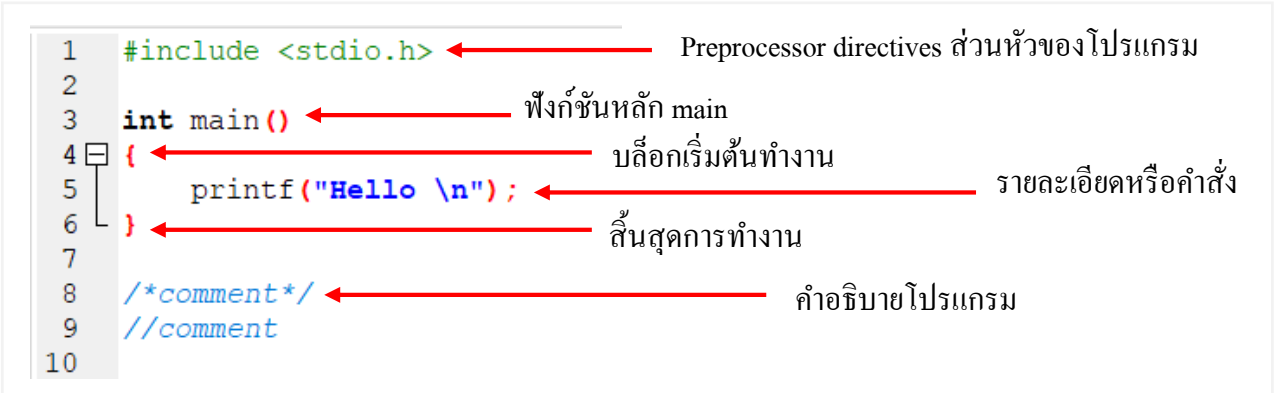
บทที่ 2 โครงสร้างภาษาซี

คำสั่งที่ใช้งานในภาษา C นั้นล้วนเป็นฟังก์ชั่นทั้งสิ้น ดังนั้นโปรแกรมที่เขียนขึ้นจึงประกอบไปด้วยฟังก์ชั่นมากมาย ที่ถูกกำหนดให้ทำหน้าที่ใดหน้าที่หนึ่งในลักษณะของโมดูลย่อย เพื่อทำงานให้บรรลุเป้าหมาย และในเมื่อภาษา C คือ ภาษาที่ประกอบไปด้วยฟังก์ชั่น

ฟังก์ชั่น (Function) คือ ชุดคำสั่งที่เขียนขึ้นเพื่อสั่งให้คอมพิวเตอร์ทำงาน ที่อนุญาตให้สามารถรับข้อมูล (Input) ประมวลผล (Processes) และแสดงผลข้อมูล (Output) โดยฟังก์ชั่นที่ถูกเขียนขึ้นใช้งาน และสามารถเรียกมาใช้งานได้ทันที จะถูกจัดเก็บไว้ในไลบารีมาตรฐาน (Standard Library) ในขณะที่ฟังก์ชั่นอื่นๆ จะเป็นฟังก์ชั่นที่ถูกเขียนขึ้นโดยโปรแกรมเมอร์ อย่างไรก็ตามในภาษา C จะมีฟังก์ชั่นพิเศษฟังก์ชั่นหนึ่งที่จำเป็นต้องมีไว้ในโปรแกรมเสมอ คือ ฟังก์ชั่น main() ทั้งนี้ฟังก์ชั่นดังกล่าวจัดเป็นฟังก์ชั่นหลักที่นำมาใช้เป็นจุดเริ่มต้นของโปรแกรมเพื่อสั่งให้ทำงาน โดยฟังก์ชั่นอื่นๆจะถือเป็นรูทีนย่อย (Subroutines)

ลักษณะโครงสร้างภาษาซีแบ่งออกได้เป็น 4 ส่วน ดังนี้

1. ส่วนประมวลผลก่อน (Preprocessor directives)
2. ส่วนฟังก์ชันหลัก (the main () function)
3. ประโยคคำสั่ง (compound statement)
4. ส่วนอธิบายโปรแกรม (program comment)



1. ส่วนประมวลผลก่อน (Preprocessor directives)

ตัวประมวลผลก่อน หรือพรีโปรเซสเซอร์ไดเร็กทีฟ ถือเป็นส่วนสำคัญส่วนหนึ่งของภาษาซี เรียกว่าส่วนหัวของโปรแกรม จำเป็นต้องกำหนดไว้ในตอนต้นโปรแกรมเสมอ ถ้าตัวแปลภาษาซีตรวจพบว่ามีการใช้พรีโปรเซสเซอร์ไดเร็กทีฟ ก็จะถูกแปลความหมายเป็นอันดับแรกก่อนคำสั่งประเภทอื่น การเขียนคำสั่งพรีโปรเซสเซอร์ไดเร็กทีฟ ต้องขึ้นต้นด้วยเครื่องหมาย #

พรีโปรเซสเซอร์จะมีอยู่หลายตัวด้วยกัน เช่น

#if	#ifdef	#ifndef	#else	#elif	#endif
#include	#define	#undef	#line	#error	#pragma

แต่พรีโปรเซสเซอร์ไดเร็กทีฟ ที่น่าสนใจและใช้งานเป็นประจำคือ **#include** และ **#define**

- พรีโปรเซสเซอร์ไดเร็กทีฟ #define เป็นการกำหนดค่านิพจน์ต่างๆ ให้กับชื่อของตัวแปร เช่น

define START 0 หมายถึง เป็นการกำหนดค่าคงที่ให้กับตัวแปร START โดยให้มีค่าเป็น 0

define temp 37 หมายถึง เป็นการกำหนดให้ตัวแปร temp มีค่าเท่ากับ 37

- พรีโปรเซสเซอร์ไดเร็กทีฟ #include ใช้สำหรับสั่งให้ตัวแปลภาษาซี นำ header file (ไฟล์ที่ระบุชื่อต่อจาก #include) มารวมกับโปรแกรมก่อนที่จะแปลภาษา สามารถเขียนได้ 2 รูปแบบดังนี้

<u>แบบที่ 1</u> #include <header file> , **<u>แบบที่ 2</u>** #include “header file”

เช่น #include<stdio.h> โดยคอมไพเลอร์จะค้นหาเฮดเดอร์ไฟล์ที่ระบุจากไดเร็กทอรี่ที่เก็บเฮดเดอร์ไฟล์โดยเฉพาะ โดยชื่อเฮดเดอร์ไฟล์ที่ผนวกเข้ามาสามารถเขียนอยู่ภายในเครื่องหมาย < > หรือ " " ก็ได้

เฮดเดอร์ไฟล์ (header file)

เฮดเดอร์ไฟล์ (header file) เป็นไฟล์ส่วนหัวที่มีการประกาศรายละเอียดของคำสั่งในภาษา C หลายคำสั่งเอาไว้ให้ผู้เขียนโปรแกรมเรียกใช้ได้ header file จะมีนามสกุล .h เช่น stdio.h เป็นต้น แต่ละไฟล์จะเก็บการประกาศของคำสั่งสำหรับใช้งานด้านต่างๆ ไม่เหมือนกัน การ include ไฟล์ .h ให้ประกาศไว้ที่ด้านบนของโปรแกรมและเราสามารถ include ไฟล์ .h เข้ามาหลายไฟล์ในโปรแกรมเดียวกันได้เช่น

เฮดเดอร์ไฟล์ที่เรานิยมใช้บ่อยมีอยู่ 2 เฮดเดอร์ คือ stdio.h และ conio.h

Header File ที่สำคัญ

stdio.h กลุ่มฟังก์ชันที่ใช้งานด้าน อินพุตและเอาต์พุต เช่น printf( ) , scanf ( ) เมื่อโปรแกรมมีการเรียกใช้งานฟังก์ชัน printf( ) ก็จะต้องผนวกเฮดเดอร์ไฟล์ stdio.h มิฉะนั้นจะคอมไพล์โปรแกรมไม่ผ่าน ภาษาซีสามารถผนวกเฮดเดอร์ได้หลาย

conio.h กลุ่มฟังก์ชันที่ใช้ควบคุมการแสดงผล, รับค่าจากคีย์บอร์ด เช่น getch ( )

string.h กลุ่มฟังก์ชันที่ใช้จัดการข้อความเช่น รวมข้อความ, ค้นหาอักขระ

math.h กลุ่มฟังก์ชันด้านการคำนวณทางคณิตศาสตร์ เช่น sin, cos, tan, log เป็นต้น

2. ฟังก์ชั่นหลัก (Main Function)

ฟังก์ชั่น main( ) ในภาษาซี จัดเป็นฟังก์ชันที่ทำหน้าเสมือนกับเป็นโปรแกรมหลักที่สั่งให้ชุดคำสั่งทำงานรวมถึงการเรียกใช้ฟังก์ชั่นย่อยๆ อื่นทำงาน กล่าวคือการสั่งงานในโปรแกรมจะต้องอยู่ภายในฟังก์ชั่น main ( ) เท่านั้น ตามด้วยเครื่องหมายปีกกาเปิด { และจบด้วยเครื่องหมายปีกกาปิด } และคำสั่งทุกคำสั่งในภาษาซีจะต้องปิดท้ายด้วยเครื่องหมาย ; (semicolon) เสมอ

ฟังก์ชัน main() สามารถเขียนในรูปแบบของ int main ก็ได้ มีความหมายเหมือนกัน คือ หมายความว่า ฟังก์ชัน main() จะไม่มีอาร์กิวเมนต์ (argument) คือไม่มีการรับค่าใด ๆ เข้ามาประมวลผลภายในฟังก์ชัน และจะมีการคืนค่ากลับออกไปจากฟังก์ชันด้วย

argument คือ ตัวรับค่าเข้ามาในฟังก์ชัน

parameter คือ ค่าที่ส่งไปยังฟังก์ชันค่า argument และ parameter ต้องเป็นข้อมูลชนิดเดียวกัน เช่น หากกำหนดให้ argument เป็นข้อมูลชนิดตัวอักษรแล้วค่า parameter ที่ส่งไปก็ต้องเป็นชนิดตัวอักษรด้วย

3. ประโยคคำสั่ง (Compound Statement)

เป็นชุดคำสั่งที่บรรจุอยู่ในฟังก์ชั่นนั่นๆ ซึ่งอาจจะเป็น

- ประโยคที่ใช้สำหรับประกาศตัวแปร (Variable) หรือการกำหนดค่าเริ่มต้นให้กับตัวแปรต่างๆ โดยตัวแปรที่ใช้งานในโปรแกรม จำเป็นต้องได้รับการประกาศชนิดข้อมูลของตัวแปรนั้นๆ ด้วย
- ประโยคนิพจน์คณิตศาสตร์ เช่น ประโยคคำนวณตัวเลขต่างๆ
- ประโยคคำสั่งควบคุมอื่นๆ เช่น คำสั่งควบคุมวงจรลูป คำสั่งควบคุมเงื่อนไข เป็นต้น

ประโยคคำสั่งเหล่านี้จะถูกบรรจุภายในบล็อก {} เพื่อใช้เป็นจุดเริ่มต้นและจุดสิ้นสุดของการทำงานในบล็อกนั้นๆ นอกจากนี้แล้ว ภายในบล็อกยังสามารถมีบล็อกย่อยๆ ซ้อนลงไปได้อีก เมื่อสิ้นสุดประโยคใดๆ จะต้องลงท้ายด้วยเครื่องหมาย ; (Semicolon)

4. คำอธิบายในโปรแกรม (Program Comment)

คำอธิบายในโปรแกรม (program comment) คือข้อความที่แทรกอยู่ภายในโปรแกรม ซึ่งคอมพิวเตอร์จะไม่แปลข้อความนี้ให้เป็นส่วนหนึ่งของโปรแกรม กล่าวคือจะไม่มีผลต่อการทำงานของโปรแกรม เขียนไว้เพื่ออธิบายโปรแกรม ซึ่งในบางครั้งผู้เขียนโปรแกรม อาจต้องการเขียนคำอธิบาย กำกับขั้นตอนการทำงานของโปรแกรมในแต่ละขั้น ทำให้เกิดประโยชน์กับผู้เขียนและผู้ที่อ่านโปรแกรมได้เข้าใจง่ายขึ้น และช่วยทำให้การแก้ไขและปรับปรุงโปรแกรมเป็นไปได้ง่ายยิ่งขึ้น สามารถเขียนได้ 2 รูปแบบ

/* และปิดด้วยเครื่องหมาย */ เป็นการระบุคำอธิบายที่ครอบคลุมหลายบรรทัด

// เป็นการระบุคำอธิบายในบรรทัดเดียวเท่านั้น

อักขระในภาษาซี

ภาษา C ได้เตรียมกลุ่มอักขระต่างๆ ให้ใช้งานเรียกว่า Character Sets แบ่งออกเป็น 2 กลุ่มใหญ่ๆ คือ กลุ่มอักขระพื้นฐาน (Basic Character set) และ (Execution Character set)

1. กลุ่มอักขระพื้นฐาน (Basic Character set) ประกอบด้วยกลุ่มอักขระ ดังต่อไปนี้

- อักษรตัวพิมพ์ใหญ่ (Alphabets Upper Case) ประกอบด้วยอักษร A-Z จำนวน 26 ตัว ดังนี้

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- อักษรตัวพิมพ์เล็ก (Alphabets Lower Case) ประกอบด้วยอักษร a-z จำนวน 26 ตัว ดังนี้

a b c d e f g h i j k l m n o p q r s t u v w x y z

- ตัวเลข (Decimal Digits) ประกอบไปด้วยตัวเลข 0 - 9 จำนวน 10 ตัว ดังนี้

0 1 2 3 4 5 6 7 8 9

- ตัวอักขระแบบกราฟิก ประกอบด้วยสัญลักษณ์ต่าง จำนวน 29 ตัว ดังนี้

, (comma)	((left parenthesis)
; (semicolon)	) (right parenthesis)
. (period)	[(left bracket)
? (question mark)	] (right bracket)
! (exclamation)	{ (left brace)
: (colon)	} (right brace)
+ (plus)	< (less than)
- (minus)	> (greater than)
* (asterisk)	= (equal sign)
/ (slash)	& (ampersand)
\ (backslash)	% (percent sign)
| (vertical bar)	# (number sign)
' (single quote)	^ (caret)
" (double quote)	_ (under score)

- ตัวอักขระแบบช่องว่าง (White space Character) ประกอบด้วยอักขระที่เป็นช่องว่างในลักษณะต่างๆ จำนวน 5 ตัว ดังนี้

blank space, horizontal tab, vertical tab, newline, form feed

2. Execution Character set ประกอบด้วยกลุ่มอักขระ ดังต่อไปนี้

-อักขระที่เป็นค่าว่าง (Null Character) อักขระที่เป็นค่าว่าง หรือค่า Null จะใช้สัญลักษณ์ \0

-อักขระควบคุม (Escape Sequence) เป็นรหัสที่ใช้ควบคุมการแสดงผลทางจอภาพและเครื่องพิมพ์ ประกอบด้วย

\a	alert (bell)	\\	backslash
\b	backspace	\?	backslash
\f	form feed	\'	single quote
\n	new line	\"	double quote
\r	carriage return	\ooo	octal number
\t	horizontal tab	\xhh	hexadecimal number
\v	vertical tab		

กฎเกณฑ์การเขียนภาษาซี

กฎเกณฑ์การเขียนภาษาซี สามารถสรุปได้ดังนี้

1. จะต้องกำหนดพรีโปรเซสเซอร์ที่ต้นโปรแกรมก่อน เช่น #include<stdio.h> , #include<conio.h>

2. คำสั่งต่าง ๆ จะต้องเขียนด้วยตัวอักษรภาษาอังกฤษพิมพ์เล็ก

3. การใช้ตัวอักษรภาษาอังกฤษพิมพ์เล็กและพิมพ์ใหญ่ในภาษาซีจะมีความแตกต่างกัน

4. ตัวแปรที่นำมาใช้งานในโปรแกรม จะต้องมีการประกาศไว้เสมอ

5. ภายในโปรแกรมต้องมีอย่างน้อยหนึ่งฟังก์ชั่น คือ main ( )

6. ใช้เครื่องหมาย { เพื่อบอกจุดเริ่มต้นของชุดคำสั่ง และเครื่องหมาย } เพื่อบอกจุดสิ้นสุดของชุดคำสั่ง โดยสามารถซ้อนเครื่องหมาย { } เพิ่มไว้ภายในได้

7. สิ้นสุดของแต่ละประโยคคำสั่ง จะต้องจบด้วยเครื่องหมาย ; (semicolon) เช่น a = b+c ;

8. สามารถใช้เครื่องหมาย /*comment*/ หรือ //comment เพื่อระบุหมายเหตุภายในโปรแกรม โดยคำอธิบายที่อยู่ภายใต้เครื่องหมาย /*comment*/ หรือ //comment จะไม่ถูกนำไปประมวลผล

หน่วยการเรียนรู้ที่ ๓ ตัวแปร ชนิดข้อมูล และตัวดำเนินการ

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๑ เข้าใจแนวคิดหลักของเทคโนโลยีเพื่อการดำรงชีวิตในสังคมที่มีการเปลี่ยนแปลงอย่างรวดเร็ว ใช้ความรู้และทักษะทางด้านวิทยาศาสตร์ คณิตศาสตร์ และศาสตร์อื่น ๆ เพื่อแก้ปัญหาหรือพัฒนางานอย่างมีความคิดสร้างสรรค์ด้วยกระบวนการออกแบบเชิงวิศวกรรม เลือกใช้เทคโนโลยีอย่างเหมาะสมโดยคำนึงถึงผลกระทบต่อชีวิต สังคม และสิ่งแวดล้อม

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๑ ม.๔/๑	วิเคราะห์แนวคิดหลักของเทคโนโลยี ความสัมพันธ์กับศาสตร์อื่น โดยเฉพาะวิทยาศาสตร์หรือคณิตศาสตร์ รวมทั้งประเมินผลกระทบที่จะเกิดขึ้นต่อมนุษย์ สังคม เศรษฐกิจ และสิ่งแวดล้อม เพื่อเป็นแนวทางในการพัฒนาเทคโนโลยี
ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์
ว ๔.๑ ม.๒/๒	ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความเข้าใจเกี่ยวกับตัวแปรในภาษาซี

๒. เรียนรู้และทำความเข้าใจเกี่ยวกับตัวดำเนินการและนิพจน์ในภาษาซี

๓. ตัวอย่างการเขียนโปรแกรมภาษาซีเบื้องต้น

<u>สาระการเรียนรู้</u>

- **ความรู้**

๑. ความรู้เกี่ยวกับตัวแปรในภาษาซี

๒. ความรู้เกี่ยวกับตัวดำเนินการและนิพจน์ในภาษาซี

๓. เขียนโปรแกรมโดยใช้ตัวแปร ตัวดำเนินการและนิพจน์

- ทักษะ / กระบวนการ

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- คุณลักษณะที่พึงประสงค์

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 3 ตัวแปร ชนิดข้อมูล และตัวดำเนินการ

ตัวแปร (Variable)

ตัวแปร (Variable) คือ การจองพื้นที่ในหน่วยความจำของคอมพิวเตอร์สำหรับเก็บข้อมูลที่ต้องใช้ในการทำงานของโปรแกรม โดยมีการตั้งชื่อเรียกหน่วยความจำในตำแหน่งนั้นด้วย เพื่อความสะดวกในการเรียกใช้ข้อมูล ถ้าจะใช้ข้อมูลใดก็ให้เรียกผ่านชื่อของตัวแปรที่เก็บเอาไว้

กฎการตั้งชื่อตัวแปร

การตั้งชื่อตัวแปรและค่าคงที่ในภาษา C จำเป็นต้องคำนึงถึงกฎเกณฑ์ต่างๆ ดังต่อไปนี้

1. ต้องขึ้นต้นด้วยตัวอักษร A-Z หรือ a-z หรือเครื่องหมาย _(Underscore) เท่านั้น

2. ภายในชื่อตัวแปรสามารถใช้ตัวอักษร A-Z หรือ a-z หรือตัวเลข 0-9 หรือเครื่องหมาย _ ได้

3. ภายในชื่อตัวแปร ห้ามเว้นช่องว่าง หรือใช้สัญลักษณ์นอกเหนือจากข้อ 2

4. ตัวอักษรพิมพ์เล็กและตัวพิมพ์ใหญ่มีความหมายแตกต่างกัน

5. ควรตั้งชื่อตัวแปรให้สื่อความหมาย

6. สามารถตั้งชื่อได้ยาวไม่จำกัด แต่จะใช้ตัวอักษรแค่ 31 ตัวแรกในการอ้างอิง

7. ห้ามตั้งชื่อซ้ำกับคำสงวน (Reserved Word)

คำสงวน หมายถึง คำที่สงวนไว้สำหรับเรียกใช้ตามวัตถุประสงค์ที่กำหนดโดยเฉพาะ เช่น คำที่ใช้ในคำสั่งควบคุม และชนิดของข้อมูล เป็นต้น ตัวอย่างคำสงวนของภาษาซี มีดังต่อไปนี้

auto	default	float	register	struct
break	do	for	return	switch
case	double	goto	short	typefef
char	else	if	signed	union
const	enum	int	sizeof	void
continue	extern	long	static	white

หากต้องการตั้งชื่อตัวแปรให้ตรงกับคำสงวน ก็สามารถทำได้โดยการนำอักษรตัวพิมพ์ใหญ่ หรือเครื่องหมาย _ เข้ามาร่วม เช่น auto → Auto , int → _int เป็นต้น

ตัวอย่างของชื่อตัวแปร ที่สามารถนำมาใช้ตั้งชื่อได้ ยกตัวอย่างเช่น

n	month	MONTH	_var1	length	EmpID
days_in_year	Do	S001	first_one	salary	num1

ตัวอย่างของชื่อตัวแปร ที่ไม่สามารถนำมาใช้เป็นชื่อตัวแปรได้ ยกตัวอย่างเช่น

int	do	123num	*variable	float	1 data

การกำหนดชนิดของตัวแปร

การเขียนโปรแกรมคอมพิวเตอร์นั้น มีความจำเป็นอย่างยิ่งที่จะต้องจัดการกับข้อมูลประเภทต่าง ๆ เพื่อให้งานนั้นๆ สามารถจัดเก็บข้อมูลได้อย่างมีประสิทธิภาพและสะดวกต่อการค้นหาข้อมูล ดังนั้นในภาษาซี จึงแบ่งประเภทของข้อมูลออกได้เป็น 6 ประเภทด้วยกันคือ

1. ข้อมูลชนิดเลขจำนวนเต็ม (Integer) คือ เลขจำนวนเต็มทั่วไป ไม่ว่าจะเป็นเลขจำนวนเต็มบวก จำนวนเต็มศูนย์และจำนวนเต็มลบ ซึ่งเลขจำนวนเต็มเหล่านี้ สามารถนำไปคำนวณได้ ตัวอย่าง เช่น 100, 56, 0, -20 เป็นต้น ซึ่งภาษาซีจะแบ่งข้อมูลชนิดนี้ออกได้เป็น 3 ประเภท คือ

- short
- int
- long

2. ข้อมูลชนิดตัวเลขทศนิยม (Float) คือ เลขทศนิยมชนิดคงที่ หรืออาจจะเป็นทศนิยม แบบไม่รู้จบ หรืออาจจะเป็นเลขทศนิยมที่เขียนในรูป E (หรือ e) ยกกำลัง ตัวเลขทศนิยมเหล่านี้ สามารถนำมาใช้ในการคำนวณได้ ตัวอย่าง เลขทศนิยมนี้ได้แก่ 20.25, -0.60, 58.96, 5.40e04 เป็นต้น แบ่งออกเป็น 3 ประเภท คือ

- float
- double
- long double

3. ข้อมูลชนิดเลขฐานแปด (Octal) คือ เลขจำนวนเต็มที่ประกอบด้วยเลข 0, 1 ,2, 3, 4, 5, 6 และ 7 เมื่อนำมาใช้ในภาษาซี จะต้องเขียนเลขศูนย์นำหน้า เช่น 0123, 045 เป็นต้น ซึ่งเลขฐานแปด เหล่านี้สามารถนำมาใช้เพื่อการคำนวณได้

4. ข้อมูลชนิดเลขฐานสิบหก (Hexadecimal) คือ ตัวเลขประเภทหนึ่งที่ใช้ในระบบคอมพิวเตอร์ ซึ่งประกอบด้วย 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e และ f เวลาใช้งานในภาษาซีจะต้องเขียนด้วย 0x นำหน้าเพื่อให้รู้ว่าตัวเลขที่นำมาใช้งานนั้นเป็นฐานสิบหก

5. ข้อมูลชนิดตัวอักขระ (Character) เป็นตัวอักษร หรือสัญลักษณ์อื่น ๆ ที่มีความยาว เพียง 1 ตัวอักษรเท่านั้น ซึ่งอาจจะเป็นตัวอักษร A-Z, a-z, 0-9 หรือ #, @, $ และอื่น ๆ เป็นต้น โดยจะเขียนไว้ในเครื่องหมาย ' ' (Single Quote) ตัวอักขระทั้งหมดนั้น สามารถศึกษาหรือดู รายละเอียดเพิ่มเติมได้จากตารางรหัส ASCII

6. ข้อมูลชนิดข้อความ (String) เป็นข้อมูลแบบตัวอักษรที่มีความยาวมากกว่า 1 ตัวอักษร มาเรียงต่อกันเป็นข้อความ โดยที่ข้อความนั้นจะต้องถูกเขียนไว้ในเครื่องหมาย " " (Double Quote) ตัวอย่างเช่น "Phitsanulok", "Welcome" เป็นต้น

ตัวแปรที่ประกาศใช้งานในโปรแกรม จำเป็นต้องถูกระบุชนิดข้อมูลเพื่อให้ทราบว่าตัวแปรเหล่านั้นจัดเก็บข้อมูลชนิดใดลงไป สำหรับในภาษา C จะมีข้อมูลพื้นฐานที่นิยมใช้ คือ

ชนิดข้อมูล	ความหมาย
char	ชนิดข้อมูลแบบตัวอักษร (Character)
int	ชนิดข้อมูลแบบเลขจำนวนเต็ม (integer)
float	ข้อมูลชนิดตัวเลขจำนวนจริง(ทศนิยม) (read or floating point)
double	ข้อมูลชนิดเลขจำนวนจริง 2 เท่า (double precision float)

นอกจากนี้ ยังสามารถเพิ่มเครื่องหมายนำหน้าให้กับชนิดข้อมูลเหล่านี้ได้อีก

signed	unsigned	long	short

โดยรายละเอียดของชนิดข้อมูลแต่ละประเภท ดังนี้

ชนิดข้อมูล	ความหมาย	ขนาด (ไบต์)	ช่วงข้อมูล
char	ตัวอักขระ	1	-128 to 127
unsigned char	ตัวอักขระ ไม่รวมเครื่องหมาย	1	0 to 255
signed char	ตัวอักขระ รวมเครื่องหมาย	1	-128 to 127
int	เลขจำนวนเต็ม	2 4	-32,768 to 32,767 2,147,483,648 to 2,147,483,647
unsigned int	เลขจำนวนเต็ม ไม่รวมเครื่องหมาย	2 4	0 to 65535 0 to 4294967
unsigned long int	เลขจำนวนเต็มแบบยาว ไม่รวมเครื่องหมาย	4	0 to 65535 0 to 4294967
unsigned short int	เลขจำนวนเต็มแบบสั้น ไม่รวมเครื่องหมาย	2	0 to 65535
signed int	เลขจำนวนเต็ม รวมเครื่องหมาย	2	-32,768 to 32,767
short int	เลขจำนวนเต็มแบบสั้น	2	-32,768 to 32,767
long int	เลขจำนวนเต็มแบบยาว	4	2,147,483,648 to 2,147,483,647
signed short int	เลขจำนวนเต็มแบบสั้น รวมเครื่องหมาย	2	-32,768 to 32,767
signed long int	เลขจำนวนเต็มแบบยาว รวมเครื่องหมาย	4	2,147,483,648 to 2,147,483,647
float	เลขจำนวนจริง มีทศนิยม	4	1.2E-38 to 3.4E+38 (6 decimal places)
doubled	เลขจำนวนจริง 2 เท่า	8	2.3E-308 to 1.7E+308 (15 decimal places)
long doubled	เลขจำนวนจริง 2 เท่าแบบยาว	10	2.3E-4932 to 1.1E+4932 (19 decimal places)

จากตาราง เป็นชนิดข้อมูลในภาษา C (ขนาดข้อมูลอาจไม่ตรงกัน ขึ้นอยู่กับแต่ละคอมไพเลอร์)

รูปแบบ การประกาศตัวแปร

```
data type variable name;
```

โดยที่ data_type คือ ชนิดข้อมูล , variable_name คือ ชื่อตัวแปร

ตัวอย่าง การประกาศตัวแปร

```
char a;          ประกาศตัวแปร a เก็บข้อมูลตัวอักษร 1 ตัว
char lastname[50]; ประกาศตัวแปร lastname เก็บข้อความ 50 ตัว
int num1;        ประกาศตัวแปร num ให้มีชนิดข้อมูลเป็นเลขจำนวนเต็ม
float total;     ประกาศตัวแปร total ให้มีชนิดข้อมูลเป็นเลขจำนวนจริง
```

การประกาศชนิดให้กับตัวแปร ควรคำนึงถึงความเหมาะสม ตัวอย่างเช่น หากต้องการใช้งานเลขจำนวนเต็มที่มีช่วงตัวเลขไม่กว้าง การเลือกใช้ short int จะเหมาะสมกว่า long int เพราะช่วยประหยัดหน่วยความจำ นอกจากนี้การสร้างสูตรคำนวณที่ประกอบไปด้วยตัวแปรชนิดข้อมูลที่ต่างกัน ตัวแปรที่ใช้เก็บผลลัพธ์จะต้องเป็นชนิดที่ใหญ่กว่าเสมอ เช่น นิพจน์ที่เป็น int มาคำนวณกับนิพจน์ที่เป็น float ตัวแปรที่ใช้เก็บผลลัพธ์ก็ต้องเป็น float

การกำหนดค่าให้กับตัวแปร

การกำหนดค่าให้กับตัวแปร จะต้องกำหนดให้เป็นไปตามกฎและต้องสัมพันธ์กับชนิดข้อมูลที่ประกาศไว้ เช่น หากกำหนดตัวแปรเป็นชนิดข้อมูล int ค่าที่กำหนดให้กับตัวแปรก็จะต้องเป็นเลขจำนวนเต็มเท่านั้น ไม่สามารถกำหนดเป็นเลขทศนิยมหรือข้อความได้

1. การกำหนดค่าชนิดเลขจำนวนเต็ม

1.ค่าตัวเลขจะต้องไม่มีทศนิยม

2.สามารถเป็นได้ทั้งค่าบวกและค่าลบ

3.สำหรับค่าบวกไม่จำเป็นต้องใส่เครื่องหมาย + นำหน้า

4.ห้ามใช้เครื่องหมายคอมม่า , หรือช่องว่างกำกับระหว่างตัวเลข เช่น 14,560 ซึ่งถือว่าผิด

5.ช่วงค่าตัวเลขจำนวนเต็ม จะอยู่ระหว่าง -2,147,483,647 ถึง 2,147,483,647

6.สามารถใช้ Suffix ต่อท้ายค่าได้ เช่น เลขจำนวนเต็มแบบยาว ก็จะใช้อักษร L (ตัวพิมพ์เล็กหรือใหญ่ก็ได้) ต่อท้ายค่า ส่วนค่าที่เป็น unsign ก็จะใช้อักษร U ต่อท้าย และใช้ UL ต่อท้ายค่าที่เป็น unsigned long

ตัวอย่าง การกำหนดค่าให้กับตัวแปร เพื่อจัดเก็บเลขจำนวนเต็มในรูปแบบต่างๆ

n1 = 445; ← ชนิดข้อมูลเป็น int

n2 = +557; ← ชนิดข้อมูลเป็น int

n3 = -6687; ← ชนิดข้อมูลเป็น int

n4 = 123456789L; ← ชนิดข้อมูลเป็น long int

n5 = 1234U; ← ชนิดข้อมูลเป็น unsigned int

n6 = 123456789UL; ← ชนิดข้อมูลเป็น unsigned long int

2. การกำหนดค่าให้กับตัวแปรชนิดเลขจำนวนจริง

1. ค่าตัวเลขสามารถมีจุดทศนิยมได้
2. สามารถเป็นได้ทั้งค่าบวก และลบ
3. ค่าบวกไม่จำเป็นต้องใส่เครื่องหมาย + นำหน้า
4. สามารถกำหนดค่าแบบเอ็กซ์โปแนนต์ได้ ด้วยการใช้อักษร E ต่อท้ายค่า
5. ค่าที่ถูกกำหนอเป็นค่าเอ็กซ์โปรแนนต์ สามารถเป็นได้ทั้งค่าบวกและลบ
6. ช่วงของค่าตัวเลขชนิดจำนวนจริงเป็นไปตามชนิดข้อมูล ซึ่งอาจถูกกำหนดเป็น float, double หรือ long double
7. สำหรับค่าจำนวนจริงชนิด double จะใช้ F ต่อท้ายค่า และใช้ L ต่อท้ายค่าที่กำหนดชนิดข้อมูลเป็น long double

ตัวอย่าง การกำหนดค่าให้กับตัวแปร เพื่อจัดเก็บเลขจำนวนเต็มในรูปแบบต่างๆ

n1 = 40.9; ← ชนิดข้อมูลเป็น float

n2 = +3.2E-5; ← ชนิดข้อมูลเป็น float exponent

n3 = 4.3E8; ← ชนิดข้อมูลเป็น float exponent

n4 = -0.2E+3 ← ชนิดข้อมูลเป็น float exponent

n5 = 12.35F ← ชนิดข้อมูลเป็น double

n6 = 12.34L; ← ชนิดข้อมูลเป็น long double

3. การกำหนดค่าชนิดตัวอักษร

1. ค่าที่กำหนดจะต้องอักขระเพียงค่าเดียว และจะต้องอยู่ภายในเครื่องหมาย ' ' ไม่ใช่เครื่องหมาย " " ดังนั้นจึงต้องพึงระมัดระวังเป็นพิเศษด้วย

2. ขนาดจำนวนตัวอักษรสูงสุด คือ 1 ตัวอักษรเท่านั้น

3. ขนาดจำนวนตัวอักษรในภาษา C จะมองเป็น ASCII ที่ใช้แทนตัวอักขระต่างๆ เช่น ค่า 'A' จะมีค่าเท่ากับ 65 หรือค่า 'a' จะมีค่าเป็น 97

ค่าคงที่ (Constants)

ค่าคงที่ (Constant) เป็นตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าในภายหลังได้ ค่าของตัวแปรค่าคงที่จะต้องถูกกำหนดพร้อมกับการประกาศตัวแปรเสมอ ค่าคงที่สามารถเป็นข้อมูลประเภทต่างๆ เหมือนประเภทข้อมูลพื้นฐานได้

ความแตกต่างระหว่างตัวแปรและค่าคงที่ได้แก่ ตัวแปรสามารถเปลี่ยนค่าไปมาได้ ในขณะที่ค่าคงที่หากถูกกำหนดค่าไว้แล้วจะไม่สามารถกำหนดค่าใหม่ได้ ตัวอย่างเช่น การกำหนดค่าคงที่แทนค่าของ Pi ซึ่งมีค่าโดยประมาณ 3.14159 คงไม่สะดวกแน่หากต้องพิมพ์ค่าตัวเลขนี้ลงไปทุกครั้งที่มีการอ้างถึง ดังนั้นในภาษา C จึงได้ประกาศค่าคงที่ขึ้นมาใช้งาน โดยต้องทำการนิยามและประกาศชื่อของตัวแทนค่าเหล่านั้น

ค่าคงที่แบ่งออกเป็น 3 ประเภท คือ 1) Literal constant 2) Defined constant 3) Memory constant

1. ค่าคงที่ที่นำมาแสดงผลโดยตรง (Literal constant)

เป็นค่าคงที่ ที่ถูกสั่งพิมพ์โดยตรง (ค่าตัวเลขหรือข้อความ) โดยไม่ได้นำค่าเหล่านั้นจัดเก็บไว้ในตัวแปร ดังนั้นค่าคงที่ประเภทนี้จึงไม่สามารถอ้างอิงหรือเรียกใช้งานซ้ำได้อีก

ตัวอย่าง การใช้คำสั่ง printf() เพื่อแสดงข้อความที่เป็นค่าคงที่แบบ Literal

```
printf ("Hello, Good morning \n");
printf ("Pitchaporn Poonsawat ");
```

ตัวอย่าง การใช้คำสั่ง printf() เพื่อแสดงข้อความและตัวเลขที่เป็นค่าคงที่แบบ Literal

```
printf ("The number value of A is %d \n" , 10);
```

2. ค่าคงที่นิยามด้วย #define (Defined Constants)

ภาษา C ได้จัดเตรียมพรีโปรเวสเซอร์ไดเร็กทีฟชื่อ #define สำหรับการกำหนดค่าคงที่โดยการประกาศใช้งานไว้ในส่วนของเฮดเดอร์ไฟล์ (Header File) โดยมีรูปแบบการประกาศใช้งานค่าคงที่ดังนี้

```
#define  ConstantsName value
```

โดยที่ #define คือ พรีโปรเวสเซอร์ไดเร็กทีฟ

ConstantsName คือ ชื่อของค่าคงที่ นิยมใช้อักษรตัวพิมพ์ใหญ่

value คือ ค่าที่ต้องการกำหนดให้ค่าคงที่

ตัวอย่าง การกำหนดค่าคงที่ด้วย #define

```
#define VAT 0.07
#define TXT “Hello”
#define NEWLINE ‘\n’
#define ONE 1
#define PI 3.141159
```

3. ค่าคงที่ที่เก็บไว้ในตัวแปร (Memory Constants)

เป็นการกำหนดค่าคงที่ในรูปแบบของตัวแปร ซึ่งจะเขียนอยู่ในฟังก์ชัน main() โดยมีรูปแบบการประกาศใช้งานค่าคงที่ดังนี้

```
const data_type variable_name = value;
```

โดยที่ const คือ การประกาศว่าตัวแปรที่กำหนดต่อไปนี้เป็นค่าคงที่

data_type คือ ชนิดข้อมูล

variable_name คือ ชื่อตัวแปรที่ใช้เก็บค่าคงที่

value คือ ค่าที่ต้องการกำหนดให้เป็นค่าคงที่

ตัวอย่าง การกำหนดค่าคงที่แบบเก็บไว้ในตัวแปร

```
const int count = 20;          //กำหนดให้ count เป็นตัวคงที่ชนิด int และเก็บค่า 20
const float PI = 3.14159;      //กำหนดให้ PI เป็นตัวคงที่ชนิด float และเก็บค่า 3.14159
```

ตัวอย่าง การกำหนดค่าคงที่และการนำไปใช้งาน

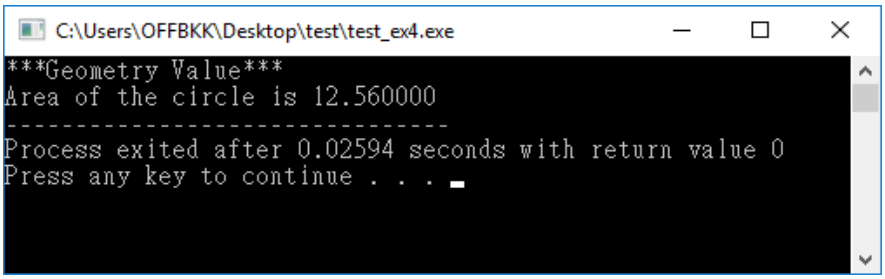
```
1   #define PI 3.14
2   #include <stdio.h>
3
4   int main()
5   {
6       int r = 2;
7       double result = 2 * PI * r;
8       printf("***Geometry Value***\n");
9       printf("Area of the circle is %f", result );
10
11  }
12
```

ค่าคงที่ที่นิยามด้วย #define

ค่าคงที่ที่เก็บไว้ในตัวแปร

ค่าคงที่ที่แสดงผลแบบโดยตรง

หน้าจอแสดงผลลัพธ์ : การหาพื้นที่วงกลม



การประกาศตัวแปร (variable declaration)

การประกาศตัวแปร คือการจองเนื้อที่ในหน่วยความจำสำหรับเก็บค่าบางอย่างพร้อมทั้งกำหนดชื่อเรียกแทนหน่วยความจำในตำแหน่งนั้น เพื่อให้สะดวกในการเข้าถึงค่าที่เก็บอยู่ในหน่วยความจำดังกล่าว

รูปแบบการประกาศค่าตัวแปร

ตัวแปรทุกๆตัวที่ใช้งานในโปรแกรม จำเป็นจะต้องประกาศก่อนใช้งานเสมอ ภาษา C สามารถประกาศตัวแปรได้หลายรูปแบบด้วยกัน คือ

1. การประกาศตัวแปรที่ละตัว ที่ละบรรทัด ตัวอย่างเช่น

```
int num;          // สร้างตัวแปรชื่อ num สำหรับเก็บข้อมูลจำนวนเต็ม
int  n ;          // สร้างตัวแปรชื่อ n สำหรับเก็บข้อมูลจำนวนเต็ม
float PI;         // สร้างตัวแปรชื่อ num สำหรับเก็บข้อมูลชนิดทศนิยม
char data;        // สร้างตัวแปรชื่อ data สำหรับเก็บข้อมูลชนิดตัวอักษร
char name[50];    // สร้างตัวแปรชื่อ name สำหรับเก็บข้อมูลชนิดตัวอักษร ขนาดไม่เกิน 50
```

2. การประกาศตัวแปรหลายๆตัว พร้อมกัน (Multiple Declaration)

เป็นการประกาศตัวแปรที่มีชนิดข้อมูลเหมือนกันภายในบรรทัดเดียวกัน โดยใช้เครื่องหมายคอมม่า , คั่นระหว่างชื่อตัวแปรแต่ละตัว ตัวอย่างเช่น

```
int lower, upper, step;
char name, surname, nickname;
```

3. การประกาศตัวแปรพร้อมกำหนดค่าเริ่มต้น

เป็นการประกาศตัวแปร พร้อมกำหนดค่าเริ่มต้นให้กับตัวแปรภายในบรรทัดนั้นๆ ตัวอย่างเช่น

```
int num = 10;        // สร้างตัวแปรชื่อ num สำหรับเก็บข้อมูลจำนวนเต็ม มีค่าเท่ากับ 10
float PI = 3.14;     // สร้างตัวแปรชื่อ PI สำหรับเก็บข้อมูลชนิดทศนิยม มีค่าเท่ากับ 3.14
char font = 'A' ;    // สร้างตัวแปรชื่อ font สำหรับเก็บข้อมูลชนิดตัวอักษร มีค่าเริ่มต้น คือ A
```

4. ประกาศตัวแปรชนิดค่าคงที่

เป็นการประกาศตัวแปรเพื่อจัดเก็บค่าคงที่ เป็นค่าในหน่วยความจำที่มีค่าคงที่ตลอดโปรแกรม ถ้าในโปรแกรมส่วนใดเรียกชื่อที่ประกาศไว้ก็จะได้ข้อมูลตามที่กำหนด จะใช้คำว่า const นำหน้า

```
const int Day = 7;
const int month = 12;
const float PI = 3.1418926;
const char name = 'A';     // ให้ค่ารหัส ASCII ของ A คือ 65
const char ch = 'B';       // ให้ค่ารหัส ASCII ของ B คือ 66
```

ตัวดำเนินการและนิพจน์

ตัวดำเนินการจะถูกใช้กับตัวแปรและค่าคงที่ในการดำเนินการบางอย่าง เช่น การดำเนินการทางคณิตศาสตร์ ในภาษา C มีตัวดำเนินการประเภทต่างๆ ที่ทำหน้าที่แตกต่างกันไป ยกตัวอย่างดังต่อไปนี้

นิพจน์ (Expression) คือ การนำค่าคงที่หรือตัวแปรมาเชื่อมต่อกัน ด้วยเครื่องหมายทางคณิตศาสตร์

โอเปอแรนด์ (Operand) หรือตัวถูกดำเนินการ จะเป็นค่าตัวเลข ซึ่งอาจเป็นได้ทั้งตัวแปรหรือค่าคที่เมื่อนำตัวดำเนินการและโอเปอแรนด์ประกอบเข้าด้วยกัน เช่น a*b/c จะเรียกว่า นิพจน์

1. ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operator**)**

ใช้สำหรับกระทำการคำนวณทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร เปอร์เซ็นต์ โดยจะนำข้อมูลตัวหนึ่งไปกระทำกับอีกตัวหนึ่ง โดยใช้ผลลัพธ์เป็นตัวเลขทางคณิตศาสตร์ แบ่งออกได้ดังต่อไปนี้

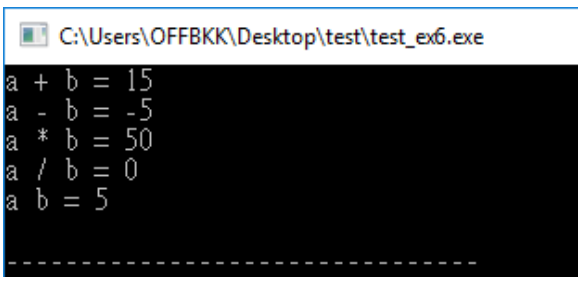
ตารางแสดงตัวดำเนินการเลขคณิต

ตัวดำเนินการ	กระบวนการ	ข้อมูลที่กระทำ	ข้อมูลผลลัพธ์	ตัวอย่าง
+	การบวก (addition)	จำนวนเต็ม,จำนวนจริง	จำนวนเต็ม,จำนวนจริง	x + y
-	การลบ (subtraction)	จำนวนเต็ม,จำนวนจริง	จำนวนเต็ม,จำนวนจริง	x - y
*	การคูณ (multiplication)	จำนวนเต็ม,จำนวนจริง	จำนวนเต็ม,จำนวนจริง	x * y
/	การหาร (division)	จำนวนเต็ม,จำนวนจริง	จำนวนจริง	x / y
%	การหารเอาเศษ (modulus)	จำนวนเต็ม	จำนวนเต็ม	x % y

ตัวอย่าง การใช้ตัวดำเนินการทางคณิตศาสตร์

```c
#include <stdio.h>

int main()
{
    int a = 5;
    int b = 10;
    printf("a + b = %d\n", a + b);
    printf("a - b = %d\n", a - b);
    printf("a * b = %d\n", a * b);
    printf("a / b = %d\n", a / b);
    printf("a % b = %d\n", a % b);
    return 0;
}
```

หน้าจอแสดงผลลัพธ์ :



2. ตัวดำเนินการยูนารี

การใช้เครื่องหมายลบนำหน้าค่าตัวแปร จะทำให้ค่าถูกเปลี่ยนเป็นค่าติดลบโดยทันที เช่น -10 , -x (มิใช่ตัวดำเนินการลบแต่อย่างใด) ตัวอย่างเช่น 100 เมื่อนำตัวดำเนินการยูนารีติดลบนำหน้าค่าดังกล่าว ค่านั้นก็จะถูกเปลี่ยนเป็นค่าติดลบทันที ซึ่งก็คือ -100

ตัวอย่าง การใช้ตัวดำเนินการยูนารี

-743 -OX7FF -0.2

-root1 -(x+y) -3*(x+y)

ตัวดำเนินการยูนารีสามารถนำมาใช้กับโอเปอแรนด์(ตัวถูกดำเนินการ) ที่เป็นค่าคงที่แบบเลขจำนวนเต็ม เลขจำนวนจริง ตัวแปร และนิพจน์ก็ได้ นอกจากนี้ในภาษา C ยังมีตัวดำเนินการยูนารีตัวอื่นๆอีก เช่น ตัวดำเนินการเพิ่มค่า (Increment Operator) ที่ใช้เครื่องหมาย ++ และตัวดำเนินการลดค่า (Decrement Operator) ที่ใช้เครื่องหมาย - - ซึ่งหมายถึง การเพิ่มค่าขึ้นทีละหนึ่งหรือการลดค่าลงทีละหนึ่ง โดยสามารถกำกับไว้หน้าตัวแปรหรือหลังตัวแปรก็ได้

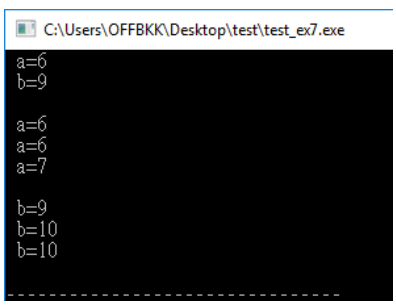
++i , i++ หมายถึง i = i + 1 , -- i , i-- หมายถึง i = i - 1

ตัวดำเนินการ	นิพจน์	ความหมาย
++ (prefix)	++a	เพิ่มค่าขึ้นอีกหนึ่งให้กับ a ก่อน แล้วจึงนำค่าใหม่ของ a ในนิพจน์นี้ไปใช้
++ (postfix)	a++	นำค่าปัจจุบันของ a ในนิพจน์นี้ไปใช้งานก่อนแล้วจึงเพิ่มค่า a ขึ้นอีกหนึ่ง
-- (prefix)	--b	ลดค่าลงอีกหนึ่งให้กับ b ก่อน แล้วจึงนำค่าใหม่ของ b ในนิพจน์นี้ไปใช้
-- (postfix)	b--	นำค่าปัจจุบันของ b ในนิพจน์นี้ไปใช้งานก่อนแล้วจึงลดค่า b ขึ้นอีกหนึ่ง

ตัวอย่าง การทำงานของตัวดำเนินการยูนารี

```c
#include <stdio.h>

int main()
{
    int a = 5;
    int b = 10;
    a++;
    b--;
    printf(" a=%d \n",a);
    printf(" b=%d \n\n",b);

    printf(" a=%d \n",a);
    printf(" a=%d \n",a++);
    printf(" a=%d \n\n",a);

    printf(" b=%d \n",b);
    printf(" b=%d \n",++b);
    printf(" b=%d \n",b);

    return 0;
}
```

หน้าจอแสดงผลลัพธ์ :



3. ตัวดำเนินการเปรียบเทียบและตรรกะ (Comparison and Logical Operators)

การเปรียบเทียบ หมายถึง การหาว่าเมื่อนำค่าที่อยู่ทางด้านซ้ายของเครื่องหมายกับค่าที่อยู่ทางขวาของเครื่องหมายเปรียบเทียบ นำมาเปรียบเทียบกันแล้วจะได้ผลลัพธ์เป็นจริงหรือเท็จ ซึ่งในแต่ละกรณีเมื่อเปรียบเทียบกันแล้ว จะได้ค่าเป็นจริงหรือเป็นเท็จเพียงค่าเดียวเท่านั้น ตัวดำเนินการเปรียบเทียบมักจะใช้กับคำสั่งตรวจสอบเงื่อนไข if และคำสั่งวนซ้ำ for while เพื่อควบคุมการทำงานของโปรแกรม

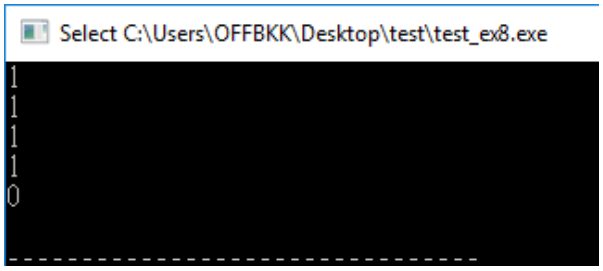
ตัวดำเนินการ	ความหมาย	ตัวอย่าง
<	น้อยกว่า	a < b
<=	น้อยกว่าหรือเท่ากับ	a <= b
>	มากกว่า	a > b
>=	มากกว่าหรือเท่ากับ	a >= b
==	เท่ากับ	a == b
!=	ไม่เท่ากับ	a != b

ผลลัพธ์จะมี 2 กรณีคือ ถ้าผลลัพธ์ถูกต้องหรือเป็นจริง (True) จะมีค่าเป็น 1 ถ้าผลลัพธ์ผิดหรือเป็นเท็จ (False) จะมีค่าเป็น 0 ซึ่งค่าตรรกะดังกล่าวจะมีชนิดข้อมูลเป็นเลขจำนวนเต็ม

ตัวอย่าง การใช้ตัวดำเนินการเปรียบเทียบและตรรกะ

```c
#include <stdio.h>

int main()
{
    int a = 5 , b = 10 , c = 15;

    printf("%d \n",a<b);
    printf("%d \n",(a + b)>=c);
    printf("%d \n",(b + c)>(a + 20));
    printf("%d \n",c != 15);
    printf("%d \n",b == 10);

    return 0;
}
```

หน้าจอแสดงผลลัพธ์ :



นอกจากนี้ ยังสามารถนำตัวเชื่อมมาใช้ร่วมกันได้ ซึ่งประกอบด้วย

ตัวเชื่อม	ความหมาย	ตัวอย่าง
&&	และ (and)	a && b
||	หรือ (or)	a || b
!	ไม่ใช่ (not)	!a

ผลลัพธ์ที่ได้ จะเป็นไปตามค่าความเป็นจริง ดังนี้

โอเปอแรนด์		ผลลัพธ์		
a	b	a && b	a || b	!a
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

4. ตัวดำเนินการระดับบิต (Bitwise Operators)

ภาษา C ยังมีตัวดำเนินการระดับบิต เพื่อใช้จัดการข้อมูลภายในเครื่องเหมือนกับภาษาระดับต่ำ จะนำข้อมูลสองค่ามาเปรียบเทียบกัน โดยข้อมูลทั้งสองค่าจะต้องเป็นข้อมูลประเภทเดียวกัน

ตัวดำเนินการ	ความหมาย
&	บิตไวส์ AND
|	บิตไวส์ OR
^	บิตไวส์ XOR (exclusive OR)
~	คอมพลีเมนต์
>>	เลื่อนบิตไปทางขวา
<<	เลื่อนบิตไปทางซ้าย

ผลลัพธ์ที่ได้จะได้จากตัวดำเนินการระดับบิต จะเป็นไปตามตารางค่าความจริง ดังนี้

โอเปอแรนด์		ผลลัพธ์			
a	**b**	**a & b**	**a | b**	**a ^ b**	**~a**
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	1	1

5. ตัวดำเนินการกำหนดค่า (Assignment operator)

ในภาษา C มีหลายวิธีในการกำหนดค่าให้กับตัวแปร ซึ่งปกติตัวดำเนินการกำหนดค่ามักจะใช้เครื่องหมาย = (เท่ากับ) และการกำหนดค่านิพจน์ก็จะใช้เครื่องหมาย + (บวก) เช่นกัน สำหรับรูปแบบของตัวดำเนินการกำหนดค่า สามารถเขียนได้ตามรูปแบบดังนี้

```
variable_name = expression
```

โดย : variable_name หมายถึง ชื่อตัวแปร

expression หมายถึง นิพจน์ ซึ่งสามารถเป็นค่าคงที่ ตัวแปร รวมถึงสูตรทางคณิตศาสตร์ที่ซับซ้อน

ตัวอย่าง : การกำหนดค่านิพจน์ด้วยตัวดำเนินการกำหนดค่า

```
a   =   3;
x   =   y;
delta   =   0.001;
sum   =   a + b;
area   =   length  *  width;
```

ตัวอย่าง : การกำหนดค่าให้นิพจน์และผลลัพธ์ที่ได้ โดยกำหนดค่าให้ตัวแปร a เป็นข้อมูลเลขจำนวนเต็ม

นิพจน์	ผลลัพธ์
a = 5.5	5
a = -5.5	5
a = 10.25	10

ตัวอย่าง : การกำหนดค่าให้นิพจน์และผลลัพธ์ที่ได้

โดยกำหนดให้ a และ b มีชนิดข้อมูลเป็นเลขจำนวนเต็ม และตัวแปร b ถูกกำหนดให้มีค่าเท่ากับ 5

นิพจน์	ผลลัพธ์
a = b	5
a = b / 2	2
a = 2 * b / 2	5
a = 2 * (b / 2)	4

นอกจากนี้ สามารถกำหนดค่าให้กับตัวแปรหลายๆ ตัวในคราวเดียวกันได้ ดังตัวอย่างต่อไปนี้

- ประกาศตัวแปร a และ b โดยกำหนดตัวแปร a และ b มีค่าเท่ากับ 10

  a = b = 10;

- ประกาศตัวแปรพร้อมๆ กันหลายตัว และกำหนดค่าเริ่มต้นของตัวแปรทั้งหมดเป็น 10

  int a, b, c;

  a = b = c = 10;

✘ รูปแบบที่ผิดของประกาศตัวแปรพร้อมๆ กันหลายตัว

int a = b = c = 10;

- นิพจน์เกี่ยวกับการสะสมค่า

  total = total + sub_total ⟶ total += sub_total

6. ตัวดำเนินการเงื่อนไข (Conditional Operator)

จะนำไปใช้ในการทดสอบค่านิพจน์ทางตรรกะว่า จริงหรือเท็จ รูปแบบดังนี้

expression1? expression2 : expression3

โดยที่ expression 1 หมายถึง นิพจน์เงื่อนไขที่สร้างขึ้น

expression 2 หมายถึง นิพจน์ กรณีเป็นจริง

expression 3 หมายถึง นิพจน์ กรณีเป็นเท็จ

ตัวอย่างเช่น

result = (i < 0) ? 0 : 100;

จากตัวดำเนินการเงื่อนไขข้างต้น อธิบายได้ว่า

i มีค่าน้อยกว่า 0 จริงหรือไม่

- ถ้า i มีค่าน้อยกว่า 0 เป็นจริง, result จะถูกกำหนดค่าเป็น 0

- ถ้า i มีค่าน้อยกว่า 0 เป็นเท็จ, result จะถูกกำหนดค่าเป็น 100

ซึ่งหากสร้างเงื่อนไขด้วยประโยคคำสั่ง if ก็จะต้องเขียนเป็น

```
if (i < 0)
   result = 0;
else
   result = 100;
```

หมายความว่า ถ้า i มีค่าน้อยกว่า 0 แล้ว

กรณีเป็นจริง ตัวแปร result จะถูกกำหนดค่าเป็น 0

กรณีเป็นเท็จ ตัวแปร result จะถูกกำหนดค่าเป็น 100

8. ตัวดำเนินการบอกขนาดข้อมูล (sizeof Operator)

ตัวดำเนินการประเภทนี้ จะทำให้เราทราบว่า ชนิดข้อมูลแต่ละชนิดที่มีอยู่ในเครื่องมีขนาดเท่าใด ทั้งนี้ชนิดข้อมูลจะมีขนาดเท่าใดนั้น จะสามารถทราบได้จากตัวดำเนินการ sizeof ซึ่งมีรูปแบบการเขียนดังนี้

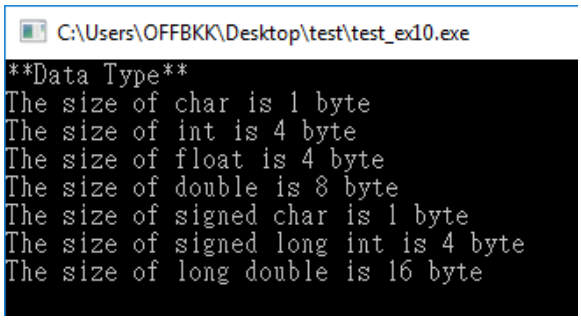
รูปแบบ : sizeof (expression)

โดย : expression หมายถึง ชนิดข้อมูลหรือตัวแปร

ตัวอย่าง : การตรวจสอบขนาดของชนิดข้อมูลประเภทต่างๆ โดยใช้ตัวดำเนินการบอกขนาดข้อมูล

```c
#include <stdio.h>

int main()
{
    printf("**Data Type**\n");
    printf("The size of char is %d byte\n", sizeof (char));
    printf("The size of int is %d byte\n", sizeof (int));
    printf("The size of float is %d byte\n", sizeof (float));
    printf("The size of double is %d byte\n", sizeof (double));
    printf("The size of signed char is %d byte\n", sizeof (signed char));
    printf("The size of signed long int is %d byte\n", sizeof (signed long int));
    printf("The size of long double is %d byte\n", sizeof (long double));
    return 0;
}
```

หน้าจอแสดงผลลัพธ์ :



หน่วยการเรียนรู้ที่ ๔ การรับและแสดงผลข้อมูล

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๒ ม.๑/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงนามธรรมเพื่อแก้ปัญหาหรืออธิบายการทำงานที่พบในชีวิตจริง
ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์
ว ๔.๒ ม.๑/๓	รวบรวมข้อมูลปฐมภูมิ ประมวลผล ประเมินผล นำเสนอข้อมูล และสารสนเทศ ตามวัตถุประสงค์ โดยใช้ซอฟต์แวร์ หรือบริการบนอินเทอร์เน็ตที่หลากหลาย
ว ๔.๒ ม.๒/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงคำนวณในการแก้ปัญหา หรือการทำงานที่พบในชีวิตจริง
ว ๔.๒ ม.๒/๒	ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความเข้าใจเกี่ยวกับฟังก์ชันที่ใช้รับข้อมูล

๒.เรียนรู้และทำความเข้าใจเกี่ยวกับฟังก์ชันที่ใช้แสดงผลข้อมูล

๓. เรียนรู้จากตัวอย่างโค้ดฟังก์ชันที่ใช้รับและแสดงผลข้อมูล

<u>สาระการเรียนรู้</u>

- **ความรู้**

๑. ความรู้เกี่ยวกับการใช้ฟังก์ชันที่ใช้รับและแสดงผลข้อมูล

๒. เขียนโปรแกรมโดยฟังก์ชันที่ใช้รับและแสดงผลข้อมูล

- ทักษะ / กระบวนการ

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- คุณลักษณะที่พึงประสงค์

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 4 การรับและแสดงผลข้อมูล (Data Input and Output)

ในการเขียนโปรแกรมต้องมีการใช้งานข้อมูล ข้อมูลต่างๆ เหล่านี้จะมีการนำไปแสดงผลและรับเข้ามาโดยผ่านอุปกรณ์อินพุตและเอ้าท์พุต เนื่องจากข้อมูลมีหลายรูปแบบตามชนิดที่มีให้ใช้ในภาษาซี ผู้เขียนโปรแกรมจึงต้องทราบวิธีการที่จะนำมาแสดงผลและรับค่าข้อมูลได้ถูกต้องตรงตามความต้องการ

การรับและแสดงผลข้อมูล (เฮดเดอร์ไฟล์ stdio.h)

ฟังก์ชันที่ใช้รับและแสดงผลข้อมูล ที่ประกาศไว้ในเฮดเดอร์ไฟล์ stdio.h ประกอบด้วย 6 ฟังก์ชันด้วยกันคือ getchar( ), putchar( ), scanf( ), printf( ), gets( ) และ puts( )

ความหมายของการแสดงผล

➤ **การแสดงผล** หมายถึง การสั่งให้คอมพิวเตอร์นำข้อมูล และผลลัพธ์ที่มีอยู่ในหน่วยความจำไปแสดงผลออกที่อุปกรณ์แสดงผล (Output Device) ของคอมพิวเตอร์ การแสดงผลที่อุปกรณ์แสดงผลอาจมีเพียงอุปกรณ์เดียว หรือหลาย ๆ อุปกรณ์พร้อมกันได้ เช่น แสดงผลที่จอภาพ เครื่องพิมพ์ ลำโพง แผ่นดิสก์

➤ **แสดงผลออกทางหน้าจอ** เป็นการทำงานพื้นฐานหรือเรียกได้ว่าเป็นส่วนหนึ่งในการทำงานของทุกโปรแกรม คือ การแสดงผลข้อมูลออกทางจอภาพ โดยในภาษาซีนั้นการแสดงผลข้อมูลออกทางจอภาพใช้คำสั่งดังนี้

การแสดงผลลัพธ์ด้วย ฟังก์ชัน printf()

ฟังก์ชันที่ใช้ในการแสดงผลข้อมูลออกทางจอภาพ คือ printf() ซึ่งเป็นฟังก์ชันที่อยู่ในไฟล์ stdio.h การเรียกใช้ฟังก์ชันนี้จะต้องทำการประมวลผลด้วยการสั่ง #include <stdio.h> ก่อนเสมอ ฟังก์ชัน printf มีหน้าที่หลักคือ แปลงข้อมูลในลักษณะของเลขฐานสอง (binary) ที่คอมพิวเตอร์ประมวลผลได้ ให้อยู่ในรูปแบบที่มนุษย์เข้าใจ ก่อนแสดงผลข้อมูลออกทางจอภาพ สามารถแสดงผลข้อมูลได้ทุกชนิดไม่ว่าจะเป็นจำนวนเต็ม (int) ทศนิยม (float) ข้อความ (string) หรืออักขระ (char) มีรูปแบบการเขียนฟังก์ชัน printf() ดังนี้

```
printf (format control, argument list);
```

โดยที่

format control คือ จะต้องเขียนอยู่ภายใต้เครื่องหมาย Double Quote “……” เป็นข้อความที่ต้องการแสดงผลออกทางหน้าจอ ซึ่งเป็นได้ทั้งข้อความธรรมดา ค่ารหัสรูปแบบข้อมูล และรหัสควบคุม

argument list คือ ค่าคงที่ ตัวแปร หรือนิพจน์ ในกรณีที่มีค่าคงที่ ตัวแปร หรือนิพจน์หลายๆ ค่าให้ใช้เครื่องหมาย , (comma) คั่นระหว่างค่าคงที่ ตัวแปร หรือนิพจน์แต่ละค่า

ตัวอย่าง การใช้งานฟังก์ชัน printf()

```
printf("Hello Program C");
printf("Do more [y/n]\n");
printf("%s","Hello....c");
printf("%s",TEXT);
printf ("1+1=%d",2);
printf("\n\n\007");
```

รหัสรูปแบบข้อมูล

รหัสรูปแบบข้อมูล ใช้สำหรับการควบคุมการแสดงผลตัวแปร หรือนิพจน์ออกทางหน้าจอ โดยรหัสรูปแบบข้อมูลมีหลายชนิดด้วยกัน การนำไปใช้จะต้องพิจารณาให้เหมาะสมกับค่าของข้อมูลที่จะแสดงผล

สำหรับรหัสรูปแบบข้อมูลที่สามารถนำมาใช้กับฟังก์ชัน print() แสดงได้ดังนี้

รหัสรูปแบบข้อมูล	ชนิดข้อมูลที่พิมพ์
%c	ตัวอักขระหนึ่งตัว (char)
%d	เลขจำนวนเต็ม (int)
%ld	เลขจำนวนเต็มแบบยาว (long)
%e	ใช้กับข้อมูลชนิดตัวเลขจุดทศนิยม (float)
%f	เลขจำนวนจริง (float)
%g	เลขจำนวนจริง (float)
%h	เลขจำนวนเต็มแบบสั้น (short integer)
%i	เลขจำนวนเต็ม (int)
%o	เลขฐานแปด
%s	ข้อความสตริง
%u	เลขจำนวนเต็ม ไม่มีเครื่องหมาย
%x	เลขฐานสิบหก

ตาราง รหัสรูปแบบข้อมูลของฟังก์ชัน printf() ซึ่งต้องสอดคล้องกับชนิดข้อมูลที่สั่งพิมพ์

นอกจากนี้ภายใน รหัสรูปแบบข้อมูลของฟังก์ชัน ยังสามารถใส่รหัสควบคุม (Escape Sequence) เข้าไปได้อีก ซึ่งรหัสควบคุมเหล่านี้จัดเป็นส่วนหนึ่งของคำสั่งควบคุมการแสดงผล ด้วยการใช้เครื่องหมาย \ (backslash) และตามด้วยรหัสควบคุม

รหัสควบคุมข้อมูล	ความหมาย
\0	ค่าว่าง (null)
\a	ส่งเสียงปี๊ป 1 ครั้ง
\b	ถอยหลังหนึ่งตัวอักษร (back space)
\f	ขึ้นหน้าใหม่ (form feed)
\n	ขึ้นบรรทัดใหม่ (new line)
\r	ย้ายเคอร์เซอร์กลับไปที่ต้นบรรทัด
\t	แท็บแนวนอน (horizontal tab)
\v	แท็บแนวตั้ง (vertical tab)
\'	พิมพ์เครื่องหมาย '
\"	พิมพ์เครื่องหมาย "
\\	พิมพ์เครื่องหมาย \

ตาราง แสดงรหัสควบคุมข้อมูล

ตัวอย่าง การใช้คำสั่ง printf() แสดงผลข้อความธรรมดาออกทางหน้าจอ

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

ผลลัพธ์ของโปรแกรม

```
1+1=2
```

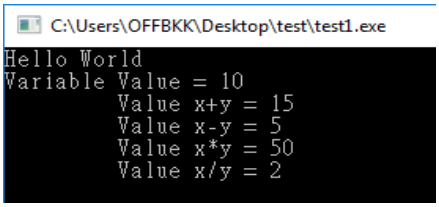
อธิบายโปรแกรม

ฟังก์ชั่น printf() ในข้อนี้มีการใช้ string format ในรูปแบบข้อความธรรมดาผสมกับรหัสรูปแบบข้อมูล และเมื่อมีรหัสรูปแบบข้อมูลแล้ว จะต้องมีการรับค่าพารามิเตอร์อีกตัวหนึ่ง สำหรับเป็นข้อมูลที่จะนำมาแทนที่ลงในตัวแทนชนิดข้อมูลนั้น จากตัวอย่าง argument list คือค่าคงที่

ตัวอย่าง การใช้คำสั่ง printf() แสดงผลข้อความธรรมดาออกทางหน้าจอ

```c
#include <stdio.h>

int main()
{
    int x=10;
    int y=5;
    printf("Hello World\n");
    printf("Variable Value = %d \n",x);
    printf("\t Value x+y = %d \n",x+y);
    printf("\t Value x-y = %d \n",x-y);
    printf("\t Value x*y = %d \n",x*y);
    printf("\t Value x/y = %d \n",x/y);
    return 0;
}
```

ผลลัพธ์ของโปรแกรม



คำอธิบายโปรแกรม

บรรทัดที่ 1	นำเฮดเดอร์ไฟล์ชื่อ stdio.h เข้ามาร่วมในการแปลผล เมื่อมีการใช้งานฟังก์ชัน printf()
บรรทัดที่ 2	ฟังก์ชัน main() ซึ่งเป็นฟังก์ชันหลักของโปรแกรม การทำงานจะเริ่มต้นที่ฟังก์ชันนี้
บรรทัดที่ 5,6	สร้างตัวแปรชนิดจำนวนเต็ม x และ y ให้มีค่า 10 และ 5 ตามลำดับ
บรรทัดที่ 7	เรียกใช้ฟังก์ชัน printf() ซึ่งเป็นฟังก์ชันมาตรฐานของภาษาซีที่ทำหน้าที่แสดงผลข้อมูลออกทางจอภาพ ในตัวอย่างจะแสดงข้อความ Hello World แล้วขึ้นบรรทัดใหม่ด้วย รหัสควบคุม \n
บรรทัดที่ 8	เรียกใช้ฟังก์ชัน printf() แสดงผลข้อความ Variable Value = 10 ออกทางจอภาพ โดยแสดงค่าตัวแปร x แทนที่ตำแหน่งของ %d ซึ่ง x มีค่าเท่ากับ 10 แล้วขึ้นบรรทัดใหม่เพราะใช้รหัสควบคุม \n
บรรทัดที่ 9	เว้นช่องว่างเป็นระยะ 1 แท็บ เมื่อเจอรหัสควบคุม \t เรียกใช้ฟังก์ชัน printf() แสดงผลข้อความ Value x+y = 15 ออกทางจอภาพ โดยแสดงผลลัพธ์ของนิพจน์ x+y (10+5) ซึ่งก็คือ 15 แทนที่ตำแหน่งของ %d แล้วขึ้นบรรทัดใหม่เพราะใช้รหัสควบคุม \n
บรรทัดที่ 10	เรียกใช้ฟังก์ชัน printf() แสดงผลข้อความ Value x-y = 5 ออกทางจอภาพ โดยแสดงผลลัพธ์ของนิพจน์ x-y (10-5) ซึ่งก็คือ 5 แทนที่ตำแหน่งของ %d แล้วขึ้นบรรทัดใหม่เพราะใช้รหัสควบคุม \n
บรรทัดที่ 11	เรียกใช้ฟังก์ชัน printf() แสดงผลข้อความ Value x*y = 50 ออกทางจอภาพ โดยแสดงผลลัพธ์ของนิพจน์ x*y (10*5) ก็คือ 50 แทนที่ตำแหน่งของ %d แล้วขึ้นบรรทัดใหม่เพราะใช้รหัสควบคุม \n
บรรทัดที่ 12	เรียกใช้ฟังก์ชัน printf() แสดงผลข้อความ Value x/y = 2 ออกทางจอภาพ โดยแสดงผลลัพธ์ของนิพจน์ x/y (10/5) ก็คือ 2 แทนที่ตำแหน่งของ %d แล้วขึ้นบรรทัดใหม่เพราะใช้รหัสควบคุม \n

การควบคุมขนาดพื้นที่แสดงผล

ตามปกติในการแสดงผลโปรแกรมจะเตรียมพื้นที่ให้พอดีกับข้อความ เช่น ถ้าจะแสดงข้อความ HELLO ซึ่งมี 5 ตัวอักษร โปรแกรมจะจัดเตรียมพื้นที่ไว้ให้พอดีสำหรับ 5 ตัวอักษร ดังแสดงรูปต่อไปนี้

H	E	L	L	O

สามารถเพิ่มพื้นที่ในการแสดงผลได้ตามต้องการ เช่น กำหนดให้แสดงข้อความ HELLO ในพื้นที่ขนาด 8 ตัวอักษร โปรแกรมจะแสดงข้อความชิดด้านขวาของพื้นที่ที่จองไว้ โดยจะเว้นพื้นที่ว่างทางด้านซ้ายอีก 3 ช่องที่เหลือเอาไว้ ดังรูป

			H	E	L	L	O

วิธีกำหนดขนาดพื้นที่การแสดงผล ให้ใส่ตัวเลขขนาดของพื้นที่ที่ต้องการไว้หลังเครื่องหมาย % ในรหัสรูปแบบข้อมูล ดังแสดงตัวอย่างต่อไปนี้

ตัวอย่าง การจัดขนาดพื้นที่การแสดงผล ด้วยฟังก์ชั่น printf() ในลักษณะต่างๆ

```c
#include <stdio.h>
main()
{
    printf("%d\n",100);
    printf("%8d\n",100);
    printf("%s\n","Test");
    printf("%9s\n","Test");
    printf("%c\n",'A');
    printf("%5c\n",'A');
    printf("%d\n",5000);
    printf("%10d\n",5000);
    printf("%f\n",3.5);
    printf("%10.2f\n",3.5);

}
```

1	0	0							
					1	0	0		
T	e	s	t						
					T	e	s	t	
A									
				A					
5	0	0	0						
						5	0	0	0
3	.	5	0	0	0	0	0		
						3	.	5	0

ผลลัพธ์ของโปรแกรม

```
C:\Users\OFFBKK\Desktop\test\test_ex13.exe
100
     100
Test
     Test
A
    A
5000
      5000
3.500000
      3.50
```

รับข้อมูลจากคีย์บอร์ดด้วยคำสั่ง scanf()

ฟังก์ชั่น scanf ( ) เป็นฟังก์ชันการรับข้อมูลจากคีย์บอร์ดมาแสดงบนหน้าจอ โดยข้อมูลที่รับเข้ามาสามารถเป็นตัวแปรชนิดตัวเลข ตัวอักขระหนึ่งตัว หรือข้อความสตริงได้ มีรูปแบบดังนี้

```
scanf(format control, &argument list);
```

โดยที่

format control คือ การใช้รหัสรูปแบบข้อมูล เพื่อกำหนดชนิดของข้อมูลที่จะรับเข้ามาจากคีย์บอร์ด โดยรหัสรูปแบบข้อมูล จะใช้ชุดเดียวกับรหัสรูปแบบข้อมูลของคำสั่ง printf()

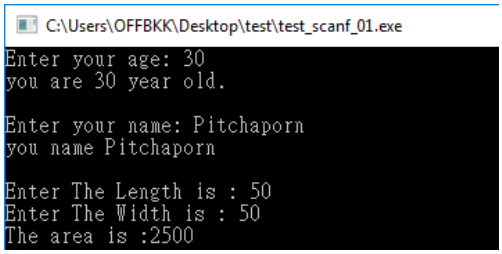
argument list คือ ตัวแปรที่จะใช้เก็บค่าข้อมูลที่รับเข้ามาจากคีย์บอร์ด โดยชนิดของตัวแปรจะต้องสอดคล้องกับรหัสรูปแบบข้อมูลที่กำหนดไว้ นอกจากนี้หน้าชื่อของตัวแปรจะต้องนำหน้าด้วยเครื่องหมาย & ยกเว้นตัวแปรสตริงสำหรับเก็บข้อความเท่านั้นที่ไม่ต้องนำหน้าด้วยเครื่องหมาย &

จำเป็นต้องจับคู่ระหว่างรหัสรูปแบบข้อมูล กับชนิดข้อมูลให้ถูกต้อง เช่น %d ใช้กับตัวแปรชนิด int

ตัวอย่าง การใช้คำสั่ง scanf() เพื่อรับค่าข้อมูล

```c
#include<stdio.h>
int main()
{
    int age;
    printf("Enter your age: ");
    scanf("%d",&age);
    printf("you are %d year old.\n\n",age);

    char name[50];
    printf("Enter your name: ");
    scanf("%s",name);
    printf("you name %s \n\n", name);

    int x,y,sum;
    printf("Enter The Length is : ");
    scanf ("%d",&x);
    printf("Enter The Width is : ");
    scanf ("%d",&y);
    sum = x*y;
    printf("The area is : %d ",sum);
    return 0;
}
```

ผลลัพธ์ของโปรแกรม



ฟังก์ชั่น scanf() เมื่ออ่านพบช่องว่างจะตัดข้อความนั้นออก

อธิบายโปรแกรม

```
int age;
printf("Enter your age: ");
scanf("%d",&age);
printf("you are %d year old.\n\n",age);
```

ประกาศตัวแปร age(อายุ) ให้มีชนิดข้อมูลเป็นจำนวนเต็ม(int)

ใช้ฟังก์ชัน printf() แสดงข้อความ Enter your age :

ใช้ฟังก์ชัน scanf() รับค่า age จากแป้นพิมพ์ ใส่ & นำหน้าตัวแปร (พิมพ์เสร็จแล้วกด Enter)

ใช้รหัสรูปแบบ %d แสดงว่าข้อมูลที่รับเข้ามาเป็นเลขจำนวนเต็ม ซึ่งจะถูกเก็บไว้ที่ตัวแปร age

ใช้ฟังก์ชัน printf() แสดงข้อความ Enter your age : และค่าอายุที่รับเข้ามา

โดยแสดงค่าตัวแปร age แทนที่ตำแหน่งของ %d (รหัสรูปแบบข้อมูลแบบเลขจำนวนเต็ม)

```
char name[50];
printf("Enter your name: ");
scanf("%s",name);
printf("you name %s \n\n", name);
```

ประกาศตัวแปร name(ชื่อ) ให้มีชนิดข้อมูลแบบตัวอักษร กำหนดขนาดไม่เกิน 50 ตัวอักษร

ใช้ฟังก์ชัน printf() แสดงข้อความ Enter your name :

ใช้ฟังก์ชัน scanf() รับค่า name จากแป้นพิมพ์ สตริงไม่ต้องใส่ & นำหน้าตัวแปร

ใช้ฟังก์ชัน printf() แสดงข้อความ your name : และค่าชื่อที่รับเข้ามา

โดยแสดงค่าตัวแปร name แทนที่ตำแหน่งของ %s (รหัสรูปแบบข้อมูลแบบข้อความสตริง)

```
int x,y,sum;
printf("Enter The Length is : ");
scanf ("%d",&x);
printf("Enter The Width is : ");
scanf ("%d",&y);
sum = x*y;
printf("The area is : %d ",sum);
```

ประกาศตัวแปร x , y , sum ให้มีชนิดข้อมูลเป็นจำนวนเต็ม (int)

ใช้ฟังก์ชัน printf() แสดงข้อความ Enter The Length is : และ Enter The Length is :

ใช้ฟังก์ชัน scanf() รับค่าตัวแปร x , y จากแป้นพิมพ์ ใส่ & นำหน้าตัวแปร

ใช้ฟังก์ชัน printf() แสดงข้อความ Enter The Length is : และ Enter The Length is : และค่าที่รับเข้ามา

ประกาศใช้ตัวแปร sum เพื่อเก็บค่าผลรวมของนิพจน์ x*y

ใช้ฟังก์ชัน printf() แสดงข้อความ The area is: และค่า sum ซึ่งได้จากผลรวมของนิพจน์ x*y

โดยแสดงค่าตัวแปร x , y , sum แทนที่ตำแหน่งของ %d (รหัสรูปแบบข้อมูลแบบเลขจำนวนเต็ม)

การรับและแสดงผลข้อมูลแบบตัวอักษร(อักขระ)

การรับและการแสดงผลข้อมูลแบบตัวอักษร นอกจากจะใช้ฟังก์ชั่น scanf() และ printf() แล้ว ยังมีฟังก์ชั่นเฉพาะในการรับข้อมูลแบบตัวอักษร คือ getchar() และแสดงผลข้อมูลแบบตัวอักษร คือ putchar()

รับข้อมูลจากคีย์บอร์ดด้วยคำสั่ง getchar()

getchar() คือฟังก์ชั่นสำหรับรับข้อมูลผ่านทางแป้นพิมพ์ โดยจะรับตัวอักษร 1 ตัวเท่านั้น

รูปแบบการใช้งานฟังก์ชัน

```
getchar( );  หรือ  char_variable = getchar( );
```

getchar( ) คือ ฟังก์ชันที่ใช้รับข้อมูลเพียง 1 ตัวอักขระจากคีย์บอร์ด โดยฟังก์ชันนี้จะไม่มี argument

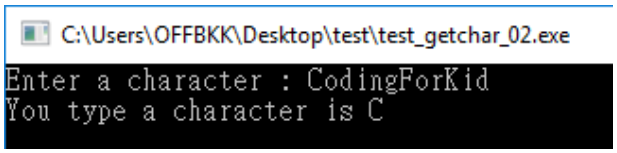
char_variable คือ ตัวแปรชนิด char ซึ่งจะเก็บข้อมูล 1 ตัวอักขระที่ป้อนผ่านทางคีย์บอร์ด

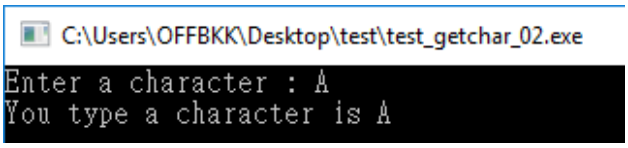
ตัวอย่าง การใช้คำสั่ง getchar( ) เพื่อรับค่าข้อมูล

```c
#include<stdio.h>
main()
{
    char ch;
    printf("Enter a character : ");
    ch = getchar( );
    printf("You type a character is %c \n",ch);
}
```

ผลลัพธ์ของโปรแกรม

เมื่อรันโปรแกรม จะเห็นข้อความ Enter a character : ปรากฏขึ้นมาพร้อมเคอร์เซอร์รอรับตัวอักษร เมื่อกรอกตัวอักษร แล้วกดปุ่ม Enter โปรแกรมจะทำงานต่อไป โดยโชว์อักขระที่รับมาเพียง 1 ตัว





อธิบายโปรแกรม

บรรทัดที่ 4 สร้างตัวแปร ch เพื่อเก็บค่าอักขระ

บรรทัดที่ 5 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter a character : ออกมาทางหน้าจอ

บรรทัดที่ 7 เรียกใช้ฟังก์ชัน getchar() เพื่อรับค่าจากคีย์บอร์ด 1 ตัวอักขระ โดยเก็บไว้ที่ตัวแปร ch

บรรทัดที่ 8 เรียกใช้ฟังก์ชัน printf() เพื่อแสดงข้อความที่กำหนด และตัวอักษรที่รับค่าจากตัวแปร ch

การแสดงผลลัพธ์ด้วย ฟังก์ชัน putchar()

คำสั่ง putchar() เป็นคำสั่งใช้สำหรับแสดงผลอักขระ โดยเฉพาะออกทางหน้าจอ มีรูปแบบการเขียนคำสั่ง ดังนี้

```
putchar(ch);
```

โดยที่

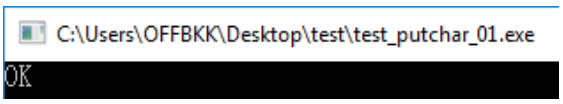
putchar( ) คือ ฟังก์ชันที่ใช้แสดงผลลัพธ์ทีละ 1 ตัวอักขระออกทางจอภาพ

ch คือ ตัวแปรชนิดอักขระ (char) ที่เขียนภายในเครื่องหมาย Single Quote (‘’)

ตัวอย่าง การใช้คำสั่ง putchar() ในการแสดงผลอักขระออกทางหน้าจอ

```
1  #include<stdio.h>
2  main()
3  {
4        char ch = 'O' ;
5        putchar(ch);
6        putchar('K');
7  }
```

ผลลัพธ์ของโปรแกรม



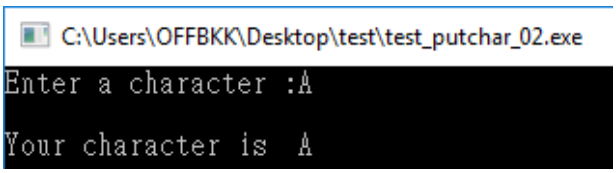
อธิบายโปรแกรม

บรรทัดที่ 4	สร้างตัวแปร ch เก็บอักษร O
บรรทัดที่ 5	เรียกใช้ฟังก์ชัน putchar() แสดงอักษร O ที่เก็บไว้ในตัวแปร first ออกมาทางหน้าจอ
บรรทัดที่ 6	เรียกใช้ฟังก์ชัน putchar() แสดงอักษร K ออกมาทางหน้าจอ

ตัวอย่าง การใช้คำสั่งฟังก์ชั่น putchar() และ getchar()

```
1  #include<stdio.h>
2  main()
3  {
4      char ch;
5      printf("Enter a character :");
6      ch = getchar();
7      printf("\nYour character is ");
8      putchar(ch);
9  }
```

ผลลัพธ์ของโปรแกรม



อธิบายโปรแกรม

บรรทัดที่ 4 สร้างตัวแปร ch เพื่อเก็บค่าอักขระ

บรรทัดที่ 5 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter a character : ออกมาทางหน้าจอ

บรรทัดที่ 6 เรียกใช้ฟังก์ชัน getchar() เพื่อรับค่าจากคีย์บอร์ด 1 ตัวอักขระ โดยเก็บไว้ที่ตัวแปร ch

บรรทัดที่ 7 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ your character is : ออกมาทางหน้าจอ

บรรทัดที่ 8 เรียกใช้ฟังก์ชัน putchar() แสดงอักษรที่รับจากคีย์บอร์ดด้วยฟังก์ชัน getchar()

รับข้อมูลจากคีย์บอร์ดด้วย ฟังก์ชัน getch ()

ฟังก์ชัน getch() เป็นอีกฟังก์ชันที่รับข้อมูลชนิดอักขระจากคีย์บอร์ดได้ครั้งละ 1 ตัวอักษรเหมือนกับฟังก์ชัน getchar() แต่การรับข้อมูลด้วยฟังก์ชัน getch() เมื่อผู้ใช้กรอกข้อมูล 1 ตัวอักษรแล้ว โปรแกรมจะทำงานต่อทันทีโดยไม่ต้องกันปุ่ม <Enter> เพียงแค่เราป้อนอักขระเข้ามา 1 ตัว โปรแกรมจะกลับไปทำงานต่อทันที และตัวอักขระที่เราป้อนจะไม่แสดงขึ้นมาให้เห็น โดยมีรูปแบบการเขียนฟังก์ชัน ดังนี้

```
getch( ); หรือ char_variable = getch( );
```

โดยที่

char_variable คือ ตัวแปรชนิด char ซึ่งจะเก็บข้อมูล 1 ตัวอักขระที่ป้อนผ่านทางคีย์บอร์ด

getch( ) คือ ฟังก์ชันที่ใช้รับข้อมูลเพียง 1 ตัวอักขระจากคีย์บอร์ด โดยฟังก์ชันนี้จะไม่มี argument ดังนั้นอาจจะใช้ getch(void) แทนคำว่า getch( ) ก็ได้ แต่นิยมใช้ getch( ) มากกว่า

ฟังก์ชั่น getch( ) ถูกประกาศไว้ในเฮดเดอร์ไฟล์ conio.h จึงต้องใส่เฮดเดอร์ไฟล์ conio.h ไว้ข้างต้น

ตัวอย่าง การใช้คำสั่ง getch() ในการรับข้อมูลจากคีย์บอร์ด

```c
#include<stdio.h>
#include<conio.h>
main()
{
    char ch;
    printf(" Enter a single character : ");
    ch = getch();
    printf("\n You type a character is %c \n",ch);
}
```

ผลลัพธ์ของโปรแกรม

```
C:\Users\OFFBKK\Desktop\test\test_getch_02.exe
Enter a single character :
You type a character is A
```

เมื่อรันโปรแกรม จะเห็นข้อความ Enter single a character : ปรากฏขึ้นมาพร้อมเคอร์เซอร์รอรับตัวอักษร เมื่อกรอกตัวอักษร เพียงตัวใดตัวหนึ่งเท่านั้น โปรแกรมก็จะทำงานต่อโดยไม่ต้องกด Enter

อธิบายโปรแกรม

บรรทัดที่ 1 นำเฮดเดอร์ไฟล์ชื่อ coino.h เมื่อมีการเรียกใช้งานฟังก์ชัน getch()

บรรทัดที่ 5 สร้างตัวแปร ch เพื่อเก็บค่าอักขระ

บรรทัดที่ 6 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter single a character : ออกมาทางหน้าจอ

บรรทัดที่ 7 เรียกใช้ฟังก์ชัน getch() เพื่อรับค่าจากคีย์บอร์ด 1 ตัวอักขระ โดยเก็บไว้ที่ตัวแปร ch

บรรทัดที่ 8 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ your type a character is : ออกมาทางหน้าจอ และแสดงอักษรที่รับจากคีย์บอร์ดด้วยฟังก์ชัน getch () จากตัวแปร ch

รับข้อมูลจากคีย์บอร์ดด้วย ฟังก์ชัน getche ( )

เป็นฟังก์ชันที่ใช้รับข้อมูลจากคีย์บอร์ดเพียง 1 ตัวอักขระ เหมือนฟังก์ชัน getch( ) แตกต่างกันตรงที่ข้อมูลที่ป้อนเข้าไป จะปรากฏให้เห็นบนจอภาพด้วย นอกนั้นมีการทำงานและลักษณะการใช้งานเหมือนฟังก์ชัน getch( ) ทุกประการ มีรูปแบบการใช้งานดังนี้

การรับข้อมูลทางคีย์บอร์ดโดยรับข้อมูลแบบตัวอักขระ1 ตัว โดยไม่ต้องกด Enter

```
getche( ); หรือ char_var = getche( );
```

การรับและแสดงผลข้อมูลแบบสตริง

การรับและการแสดงผลข้อมูลแบบสตริง นอกจากจะใช้ฟังก์ชั่น scanf() และ printf() แล้ว ยังมีฟังก์ชั่นเฉพาะในการรับข้อมูลแบบสตริง คือ gets() และแสดงผลข้อมูลแบบสตริง คือ puts()

ข้อมูลประเภทสตริง คือ กลุ่มข้อความ ซึ่งท้ายข้อความจะมีการผนวกค่า Null หรือรหัสควบคุม \0 ปะต่อท้ายโดยอัตโนมัติ เพื่อใช้บ่งบอกถึงจุดสิ้นสุดของข้อความนั้นๆ ทั้งนี้การจัดเก็บข้อความสตริงในภาษา C จะจัดเก็บในรูปแบบ Array สำหรับฟังก์ชัน gets( ) และ puts( ) ถือเป็นอีกทางเลือกหนึ่งของการนำไปใช้เพื่อการรับค่าและแสดงผล แทนที่จะใช้ฟังก์ชัน scanf( ) หรือ printf( ) เท่านั้น

รับข้อมูลจากคีย์บอร์ดด้วยฟังก์ชั่น gets()

นอกจากคำสั่งในการรับข้อมูลชนิดอักขระหรือตัวอักษรแล้ว การรับข้อความจากคีย์บอร์ดก็มีคำสั่งพิเศษให้เรียกใช้งานเช่นกัน ก็คือฟังก์ชั่น gets() มีรูปแบบดังนี้

```
gets(str);
```

โดยที่

คำสั่ง gets() เป็นอีกฟังก์ชันที่รับข้อมูลเป็นข้อความจากคีย์บอร์ด ซึ่งเป็นฟังก์ชันสำหรับข้อความโดยเฉพาะ โดยมีรูปแบบการเขียนฟังก์ชัน ดังนี้

str เป็นตัวแปรที่ใช้สำหรับเก็บข้อความ ซึ่งจะต้องสร้างเตรียมไว้ก่อนที่จะเรียกใช้งานฟังก์ชัน gets()

ตัวอย่าง การใช้ฟังก์ชั่น gets() ในการรับข้อความจากคีย์บอร์ด

```
1  #include<stdio.h>
2  main()
3  {
4      char name[20];
5      printf("Enter your name : ");
6      gets(name);
7      printf("You name is : %s",name);
8  }
```

ผลลัพธ์ของโปรแกรม

```
C:\Users\OFFBKK\Desktop\test\test_gets_01.exe
Enter your name : Pitchaporn Poonsawat
You name is : Pitchaporn Poonsawat
```

เมื่อรันโปรแกรม จะเห็นข้อความ Enter your name : ปรากฏขึ้นมาพร้อมเคอร์เซอร์รอรับข้อความ
เมื่อผู้ใช้พิมพ์ข้อความ และกด <Enter> โปรแกรมก็จะทำงานโดยแสดงผลที่หน้าจอ
ฟังก์ชั่น gets() สามารถรับค่าที่เป็นช่องว่างได้ (space) โดยรับไปจนกว่าจะกด <Enter>

อธิบายโปรแกรม

บรรทัดที่ 4	สร้างตัวแปรชนิด char ชื่อ name[20] เพื่อเก็บชนิดข้อความ ความยาว 20 ตัวอักษร
บรรทัดที่ 5	เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter your name : ออกมาทางหน้าจอ
บรรทัดที่ 6	เรียกฟังก์ชัน gets() เพื่อรับข้อความและเก็บไว้ที่ตัวแปร name โดยพิมพ์ข้อความและจะต้องกดปุ่ม Enter
บรรทัดที่ 7	เรียกใช้ฟังก์ชัน printf() เพื่อแสดงข้อความ Your name is : และข้อความที่รับค่าจากตัวแปร name

การแสดงผลลัพธ์ด้วย ฟังก์ชัน puts()

คำสั่ง puts() มาจากคำว่า puts string เป็นฟังก์ชั่นสำหรับแสดงข้อความสตริงออกทางหน้าจอ หลักการใช้งานคือส่งค่าที่อยู่ (address) ของสตริงเข้ามาที่ฟังก์ชั่น puts() โดยมีรูปแบบการเขียนคำสั่ง ดังนี้

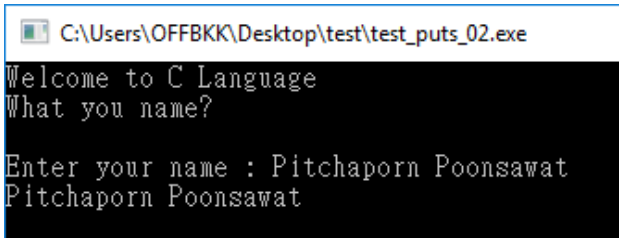
```
puts(str);
```

โดยที่ str เป็นตัวแปรที่เก็บข้อมูลชนิดข้อความ (string) ที่เขียนภายในเครื่องหมาย Double Quote (“ ”)

ตัวอย่าง การแสดงผลโดยใช้ฟังก์ชั่น gets() และ puts()

```c
#include<stdio.h>
main()
{
    char message[] = "Welcome to C Language";
    puts(message);
    puts("What you name? \n");
    char name[20];
    printf("Enter your name : ");
    gets(name);
    puts(name);
}
```

ผลลัพธ์ของโปรแกรม



อธิบายโปรแกรม

บรรทัดที่ 4 สร้างตัวแปร message เก็บข้อมูลชนิดข้อความ(char) Welcome to C Language

บรรทัดที่ 5 เรียกใช้ฟังก์ชัน puts() แสดงข้อความที่เก็บไว้ในตัวแปร message ออกมาทางหน้าจอหลังจากแสดงข้อความแล้วจะขึ้นบรรทัดใหม่ให้ด้วย

บรรทัดที่ 6 เรียกใช้ฟังก์ชัน puts() แสดงข้อความ What your name? ออกมาทางหน้าจอบรรทัดใหม่เพราะมีการใช้ \n

บรรทัดที่ 7 ประกาศตัวแปรชนิด char ชื่อ name[20] เพื่อเก็บชนิดข้อความ ความยาว 20 ตัวอักษร

บรรทัดที่ 8 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter your name? ออกมาทางหน้าจอ

บรรทัดที่ 9 เรียกใช้ฟังก์ชัน gets() เพื่อรับข้อความและเก็บไว้ที่ตัวแปร name พิมพ์ข้อความและจะต้องกดปุ่ม Enter

บรรทัดที่ 10 เรียกใช้ฟังก์ชัน puts() แสดงข้อความที่เก็บไว้ในตัวแปร name ออกมาทางหน้าจอ

หน่วยการเรียนรู้ที่ ๕ ขั้นตอนการทำงานของโปรแกรม

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๑ เข้าใจแนวคิดหลักของเทคโนโลยีเพื่อการดำรงชีวิตในสังคมที่มีการเปลี่ยนแปลงอย่างรวดเร็ว ใช้ความรู้และทักษะทางด้านวิทยาศาสตร์ คณิตศาสตร์ และศาสตร์อื่น ๆ เพื่อแก้ปัญหาหรือพัฒนางานอย่างมีความคิดสร้างสรรค์ด้วยกระบวนการออกแบบเชิงวิศวกรรม เลือกใช้เทคโนโลยีอย่างเหมาะสมโดยคำนึงถึงผลกระทบต่อชีวิต สังคม และสิ่งแวดล้อม

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๑ ม.๑/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็น นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนและดำเนินการแก้ปัญหา
ว ๔.๑ ม.๑/๔	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็น นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนและดำเนินการแก้ปัญหา
ว ๔.๒ ม.๑/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงนามธรรมเพื่อแก้ปัญหาหรืออธิบายการทำงานที่พบในชีวิตจริง
ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์
ว ๔.๒ ม.๑/๓	รวบรวมข้อมูลปฐมภูมิ ประมวลผล ประเมินผล นำเสนอข้อมูล และสารสนเทศ ตามวัตถุประสงค์ โดยใช้ซอฟต์แวร์ หรือบริการบนอินเทอร์เน็ตที่หลากหลาย
ว ๔.๒ ม.๒/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงคำนวณในการแก้ปัญหา หรือการทำงานที่พบในชีวิตจริง
ว ๔.๒ ม.๒/๒	ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. ทำความเข้าใจการการเขียนโปรแกรม เพื่อแก้ปัญหาใดปัญหาหนึ่งจนได้ผลลัพธ์

๒.เรียนรู้และทำความเข้าใจเกี่ยวกับขั้นตอนในการเขียนโปรแกรม

- การวิเคราะห์ปัญหา
- การเขียนผังงาน
- การเขียนโปรแกรม
- การทดสอบโปรแกรม
- จัดทำคู่มือ

๓. เรียนรู้และฝึกทักษะจากตัวอย่างโค้ดโปรแกรม

<u>สาระการเรียนรู้</u>

- **ความรู้**

๑. ความรู้เกี่ยวกับขั้นตอนในการเขียนโปรแกรม

๒. เขียนโปรแกรมได้ผลลัพธ์ที่ถูกต้อง

- **ทักษะ / กระบวนการ**

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 5 ขั้นตอนการทำงานของโปรแกรม

การเขียนโปรแกรมเพื่อแก้ปัญหาใดปัญหาหนึ่งจนได้ผลลัพธ์ออกมาตามต้องการนั้น หากผู้เขียนโปรแกรมมีแนวคิดในการเขียนโปรแกรมที่ดี และทำการเขียนโปรแกรมตามแนวคิดที่ได้วางไว้ แต่สำหรับผู้เขียนโปรแกรมหน้าใหม่อาจจะยังไม่เข้าใจหลักการในการเขียนโปรแกรมว่าจะต้องเริ่มยังไง ทำให้คิดว่าการเขียนโปรแกรมเป็นเรื่องยากไปเลย ดั้งนั้นในบทความนี้จะกล่าวถึงหลักการ แนวคิดการเขียนโปรแกรมเพื่อเป็นแนวทางสำหรับผู้เขียนโปรแกรมได้เข้าใจหลักการมากขึ้น

การเขียนโปรแกรม เป็นกระบวนการใช้ภาษาคอมพิวเตอร์เพื่อกำหนดโครงสร้างของข้อมูลและกำหนดขั้นตอนวิธี เพื่อใช้แก้ปัญหาตามที่ออกแบบไว้ ในการเขียนโปรแกรมนั้นมีขั้นตอนหลักๆ 5 ขั้นตอน

โจทย์ : จงเขียนโปรแกรมรับค่าเลขจำนวนเต็มทั้ง 2 ตัวแล้วจงหาผลบวกเลขทั้ง 2 จำนวนนั้น

1. วิเคราะห์ปัญหา (Analysis)

ขั้นตอนนี้เป็นขั้นตอนสำคัญที่สุด ผู้เขียนโปรแกรมต้องวิเคราะห์ปัญหาให้ออกว่าจะต้องทำการเขียนโปรแกรมเพื่อแก้ปัญหาอะไร เพราะถ้าหากวิเคราะห์ผิดแล้ว ผลลัพธ์ที่ได้ก็จะผิดไปด้วยเช่นกัน เป็นการแยกแยะรายละเอียดของปัญหาและความต้องการออกเป็นส่วนย่อยๆ ให้ครอบคลุมการทำงานของโปรแกรมที่ต้องการเขียนทั้งหมด เพื่อให้เห็นถึงองค์ประกอบ ความสัมพันธ์ ความต้องการ และแนวทางการแก้ปัญหาที่ถูกต้องอย่างครบถ้วน

จากโจทย์ข้างต้น สามารถแตกปัญหาได้เป็น 2 ส่วน

- **ต้องรับข้อมูลเลขจำนวนเต็ม 2 ตัวเข้ามาในโปรแกรม**

วิเคราะห์ : กำหนดให้ x เก็บเลขจำนวนเต็มตัวที่ 1

กำหนดให้ y เก็บเลขจำนวนเต็มตัวที่ 2

- **เลขจำนวนเต็มตัวที่ 1 + เลขจำนวนเต็มตัวที่ 2 เท่ากับเท่าไร**

วิเคราะห์ : กำหนดให้ sum เก็บค่าผลบวกของตัวเลขจำนวนเต็มทั้ง 2 จำนวน คือ

sum = x + y

วางแผนและออกแบบ (Planning & Design)

การวางแผน คือการนำปัญหาที่วิเคราะห์ได้จากขั้นตอนที่ 1 มาวางเป็นแผนอย่างเป็นขั้นเป็นตอนว่าจะต้องเขียนโปรแกรม เพื่อแก้ปัญหาอย่างไร การวางแผนอย่างเป็นขั้นตอนนี้เรียกว่า อัลกอริทึม (Algorithm) ซึ่งแบ่งออกเป็น 2 แบบหลักๆ คือ ซูโดโค้ด (Pseudo code) และโฟล์ชาร์ต (Flowchart)

อัลกอริทึม (Algorithm)

คือ กระบวนการ การทำงานที่ใช้การตัดสินใจ โดยนำหลักเหตุผลและคณิตศาสตร์มาช่วยเลือกวิธีการหรือขั้นตอนการดำเนินงานต่อไป จนกระทั่งถึงขั้นตอนสุดท้าย เป็นวิธีการที่ใช้แยกย่อยและเรียงลำดับขั้นตอนของกระบวนการในการทำงานต่างๆ เพื่อเพิ่มประสิทธิภาพในการค้นหาและแก้ไขปัญหา

คุณสมบัติของอัลกอริทึม

1. เป็นกระบวนการที่สร้างขึ้นจากกฎเกณฑ์ เนื่องจากอัลกอริทึมจัดเป็นรูปแบบหนึ่งของการแก้ปัญหา และกระบวนการก็คือกลุ่มของขั้นตอนที่อยู่ร่วมกันเพื่อใช้แก้ปัญหาต่างๆ เพื่อให้ได้มาซึ่งผลลัพธ์ ดังนั้น จึงจำเป็นต้องมีกฎเกณฑ์ในการสร้างกระบวนการแหล่านั้น ซึ่งอาจอยู่ในรูปแบบของประโยคภาษาอังกฤษ สัญลักษณ์ หรือ ซูโดโค้ด ที่มีความเป็นสากล

2. กฎเกณฑ์ที่สร้างอัลกอริทึมต้องไม่คลุมเครือ รูปแบบของกฎเกณฑ์ใดก็ตามที่ใช้สร้างอัลกอริทึมนั้น สมควรจะต้องมีระบบ ระเบียบอ่านแล้วไม่สับสน กล่าวคือ จะต้องเป็นกฎเกณฑ์ที่อ่านแล้วเข้าใจตรงกันและที่สำคัญควรหลีกเลี่ยงคำที่ก่อให้เกิดความเข้าใจได้หลายความหมาย

3. การประมวลผลต้องเป็นลำดับขั้นตอน คำสั่งต่างๆ ที่ถูกกำหนดด้วยกฎเกณฑ์จะต้องประมวลผลเป็นลำดับตามขั้นตอนที่แน่นอน

4. กระบวนการต้องให้ผลลัพธ์ตามที่กำหนดในปัญหา กลุ่มขั้นตอนต่างๆ ที่กำหนดไว้ในกระบวนการ เมื่อถูกดำเนินการแล้ว เมื่อถูกดำเนินการแล้ว จะต้องมีผลลัพธ์ตรงตามที่กำหนดในปัญหานั้นๆ

5. อัลกอริทึมต้องมีจุดสิ้นสุด คุณสมบัติอีกข้อหนึ่งที่สำคัญก็คือ อัลกอริทึมต้องมีจุดสิ้นสุด เนื่องจากคอมพิวเตอร์จะไม่สามารถประมวลผลแบบไม่สิ้นสุดได้ เช่น การบวกเลขจำนวนเต็มบวก ไม่ถือว่าเป็นอัลกอริทึม เนื่องจากไม่ทราบขอบเขตสิ้นสุดของตัวเลขจำนวนเต็ม ดังนั้น คำว่า “การบวกเลขจำนวนเต็มบวก” จึงเป็นขั้นตอนการทำงานแบบไม่มีที่สิ้นสุด

รูปแบบของอัลกอริทึม (Algorithm)

อัลกอริทึมที่เขียนขึ้นมานั้น เมื่อนำมาแสดงขั้นตอนการทำงานของโปรแกรม จะมีรูปแบบที่นำมาใช้แสดงแตกต่างกัน แบ่งออกเป็น 2 ลักษณะดังนี้

1. ซูโดโค้ด (Pseudo code)

เป็นคำอธิบายขั้นตอนการทำงานของโปรแกรม โดยใช้ถ้อยคำผสมระหว่างภาษาอังกฤษและภาษาการเขียนโปรแกรมแบบโครงสร้าง จะช่วยให้ผู้เขียนโปรแกรมสามารถพัฒนาขั้นตอนต่างๆ ให้เป็นโปรแกรมได้ง่ายขึ้น ซูโดโค้ดที่ดีจะต้องมีความชัดเจน ได้ใจความ ข้อมูลต่าง ๆ จะถูกเขียนอยู่ในรูปตัวแปร

โจทย์ : จงเขียนโปรแกรมรับค่าเลขจำนวนเต็มทั้ง 2 ตัวแล้วจงหาผลบวกเลขทั้ง 2 จำนวนนั้น

จากโจทย์ข้างต้น สามารถเขียนเป็นอัลกอริทึมรูปแบบซูโดโค้ด ได้ดังนี้

```
1. START
2. INPUT X
3. INPUT Y
4. COMPUTE SUM = X+Y
5. PRINT SUM
6. STOP
```

ตัวอย่างที่ 5.1 การเขียนซูโดโค้ด สำหรับให้คอมพิวเตอร์หาค่าเฉลี่ยจากข้อมูลที่รับเข้าทางแป้นพิมพ์ ถ้าใส่ค่า 0 แสดงว่าหยุดป้อนข้อมูล เขียนได้ดังนี้

การวิเคราะห์ข้อมูล :

1. เริ่มต้น
2. ตัวนับ = 0
3. ผลรวม = 0
4. รับค่าทางแป้นพิมพ์เก็บไว้ในตัวแปร (ข้อมูล)

5. ถ้า ข้อมูล มากกว่า 0

   เพิ่มค่าตัวนับขึ้นหนึ่งค่า

   ผลรวม = ผลรวม + ค่าข้อมูล

   ย้อนกลับไปทำขั้นตอนที่ 3

   ถ้าไม่มากกว่าไปทำขั้นตอนที่ 5

6. ค่าเฉลี่ย = ผลรวมหารด้วยตัวนับ

7. แสดงค่าเฉลี่ยทางจอภาพ (ทศนิยมสองตำแหน่ง)

8. สิ้นสุด

อัลกอริทึมรูปแบบซูโดโค้ด :

```
1. START
2. count = 0
3. sum = 0
4. INPUT (value)
5. IF value  >  0 THEN
     count = count +1
     sum = sum + value
     GOTO 3
     ELSE GOTO 5
6. average = sum / count
7. OUTPUT (average)
8. END
```

ตัวอย่างที่ 5.2 การเขียนซูโดโค้ด คำนวณหาพื้นที่สามเหลี่ยม ได้ดังนี้

การหาพื้นที่สามเหลี่ยม

1. เริ่มต้น
2. รับค่าความยาวของฐานมาเก็บในตัวแปร X
3. รับค่าความยาวของสูงมาเก็บในตัวแปร Y
4. คำนวณหาพื้นที่ ARRAY = (X*Y) / 2
5. แสดงผลพื้นที่
6. จบ

อัลกอริทึมรูปแบบซูโดโค้ด :

1. START
2. READ X
3. READ Y
4. ARRAY = (X*Y) / 2
5. PRINT ARRAY
6. END

ขั้นตอนการทำงานแบบเรียงลำดับ

สำหรับอัลกอริทึมในคอมพิวเตอร์ จะเป็นชุดการทำงานของรหัสคำสั่งทำงานที่มีการประมวลผลทีละคำสั่งเรียกว่า ประโยคคำสั่ง โดยสามารถมีจำนวนมากหลายๆ คำสั่ง เช่น ให้ทำการบวก ลบ คูณ หาร ตัวเลขในทางคณิตศาสตร์ ซึ่งเป็นปัญหาในการเขียนโปรแกรมให้ทำงาน

ตัวอย่างที่ 5.3 การรวบรวมคะแนนของผู้เรียน 3 คน เพื่อนำมาหาคะแนนเฉลี่ย โดยอธิบายถึงการรับค่าเข้ามา การหาคะแนนเฉลี่ย และการนำออกมาแสดงผล มีขั้นตอนดังต่อไปนี้

1. เริ่มต้นทำงาน
2. กำหนดให้ sum เท่ากับ 0
3. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 1 เก็บไว้ใน score
4. นำค่าจาก score มารวมเก็บไว้ใน sum
5. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 2 เก็บไว้ใน score
6. นำค่าจาก score มารวมเก็บไว้ใน sum
7. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 3 เก็บไว้ใน score
8. นำค่าจาก score มารวมเก็บไว้ใน sum
9. แสดงผลค่าเฉลี่ย โดยนำค่าที่เก็บไว้ใน sum หารด้วย 3
10. จบการทำงาน

อัลกอริทึมรูปแบบซูโดโค้ด :

```
1. START
2. SET sum = 0
3. INPUT score1
4. SUM = sum + score1
5. INPUT score2
6. sum = sum + score2
7. INPUT score3
8. sum = sum + score3
9. AVERAGE = sum/3
10. END
```

จากตัวอย่างที่ 1.3 แสดงให้เห็นขั้นตอนการทำงานแบบเรียงลำดับที่มี 10 ขั้นตอน ตั้งแต่เริ่มต้นจนสิ้นสุดโปรแกรม ต้องทำขั้นตอนหนึ่งให้เสร็จก่อน จึงจะไปทำขั้นตอนถัดไป

ขั้นตอนการทำงานแบบตัดสินใจ

จากตัวอย่างที่ 1.3 เป็นขั้นตอนการทำงานแบบเรียงลำดับ แต่ยังสามารถมีข้อผิดพลาดได้ เนื่องจากไม่มีการทำงานที่ตรวจสอบ หากค่าของคะแนนสอบที่รับเข้ามาไม่อยู่ในช่วงตั้งแต่ 0-100 จะทำให้หาค่าเฉลี่ยไม่ถูกต้อง ดังนั้นการเขียนอัลกอริทึมต้องมีขั้นตอนการทำงานที่เป็นการตัดสินใจ มีทางเลือกเพิ่มเข้ามาอยู่ในรูปแบบของเงื่อนไข ใช้ตรวจสอบว่าค่าที่รับเข้ามาเป็นจริงหรือเท็จ ซึ่งมีให้เลือกใช้คือ

ถ้า ... แล้ว (if..then) เป็นการตรวจสอบเงื่อนไข เป็นจริงเท่านั้นถึงจะให้มีการทำงาน หากเป็นเท็จให้ข้ามไปทำลำดับถัดไปทันที มีลักษณะทางเลือกเดียวคือให้เลือกทำหรือไม่ทำ

ถ้า ... แล้ว ... ไม่เช่นนั้น (if..then…else) เป็นการตรวจสอบเงื่อนไข ถ้าเป็นจริงจะให้ทำงานในส่วนแรก แต่หากเป็นเท็จจะให้ทำงานในส่วนที่สอง มีลักษณะเป็นสองทางเลือก ให้เลือกทำทางใดทางหนึ่ง

ตัวอย่างที่ 5.4 การเขียนซูโดโค้ด การทำงานแบบมีการตัดสินใจ

```
1. เริ่มต้นทำงาน
2. กำหนดให้ sum เท่ากับ 0
3. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 1 เก็บไว้ใน score
4. ถ้าค่าใน score อยู่ระหว่าง 0 -100 แล้ว
        4.1 นำไปรวมเก็บไว้ใน sum
5. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 2 เก็บไว้ใน score
6. ถ้าค่าใน score อยู่ระหว่าง 0 -100 แล้ว
        6.1 นำไปรวมเก็บไว้ใน sum
7. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 3 เก็บไว้ใน score
8. ถ้าค่าใน score อยู่ระหว่าง 0 -100 แล้ว
        8.1 นำไปรวมเก็บไว้ใน sum
9. แสดงผลค่าเฉลี่ย โดยนำค่าที่เก็บไว้ใน sum หารด้วย 3
10. จบการทำงาน
```

อัลกอริทึมรูปแบบซูโดโค้ด :

```
1. START
2. SET sum = 0
3. INPUT score1
4. IF score >= 0 AND score <= 100 THEN
        4.1 sum = sum + score1
   ENDIF
5. INPUT score2
6. IF score >= 0 AND score <= 100 THEN
        sum = sum + score2
   ENDIF
7. INPUT score3
8. IF score >= 0 AND score <= 100 THEN
        sum = sum + score3
   ENDIF
9. DISPLAY = SUM/3
10. END
```

ขั้นตอนการทำงานแบบวนซ้ำ

จากตัวอย่างที่ 5.4 มีความถูกต้องในเรื่องขั้นตอนการทำงาน และการจัดการกับข้อมูลที่เกี่ยวข้อง แต่ก็ยังขาดประสิทธิภาพ จึงต้องนำมาขัดเกลา ซึ่งจะเห็นว่าขั้นตอนที่เกี่ยวกับการรับค่าเข้ามา มีลักษณะที่เหมือนกัน 3 ชุด และเป็นการแก้ไขปัญหาแบบรูปธรรม หากมีความต้องการหาค่าเฉลี่ยจากคะแนนสอบที่มีจำนวน 10 ค่า ก็ต้องเปลี่ยนแปลงแก้ไขอัลกอริทึม ให้มีการรับค่าเข้ามาได้ 10 ชุด และอาจไม่เหมาะสมกับการเขียนโปรแกรมที่ต้องมีขั้นตอนมากขึ้นตามจำนวนการรับค่าไปด้วย

ดังนั้นเพื่อความสะดวก จึงมีการใช้ขั้นตอนการทำงานที่เป็นการวนซ้ำหรือการวนลูป โดยให้ขั้นตอนการทำงานที่เหมือนกัน ถูกใช้งานเพียงจุดเดียว และทำงานซ้ำแบบเดิมวนรอบตามจำนวนที่ต้องการ ในการเขียนอัลกอริทึมที่มีขั้นตอนการวนซ้ำ จะนำรูปแบบของเงื่อนไขมาใช้ตรวจสอบว่าเป็นจริงหรือเท็จ เพื่อให้การทำงานซ้ำแบบเดิมต่อไปหรือไม่ ซึ่งมีให้เลือกใช้คือ

ขณะที่ ... จนกระทั่ง (Repeat..Until) เป็นการทำงานซ้ำแบบเดิม ตราบใดการตรวจสอบเงื่อนไขยังเป็นเท็จ จนกระทั่งการตรวจสอบเงื่อนไขเป็นจริง จึงหยุดการวนซ้ำ

ขณะที่ ... (While) ตราบใดที่การตรวจสอบเงื่อนไขเป็นจริง การทำงานก็จะยังคงวนซ้ำแบบเดิม หากเป็นเท็จจึงจะหยุดการวนซ้ำ

ตัวอย่างที่ 5.5 การเขียนซูโดโค้ด การทำงานแบบมีการวนซ้ำ

```
1. เริ่มต้นทำงาน
2. กำหนดให้ sum เท่ากับ 0
3. ขณะที่
        3.1 แสดงข้อความการรับค่าคะแนนสอบ
        3.2 รับค่าคะแนนสอบจากผู้เรียนเก็บไว้ใน score
        3.3 ถ้าค่าใน score อยู่ระหว่าง 0 -100 แล้ว
                3.3.1 นำไปรวมเก็บไว้ในตัวแปร sum
                3.3.2 แจ้งให้ทราบว่า ค่าที่ป้อนไม่ถูกต้อง ให้ทำการป้อนใหม่
                จนกว่า รับค่าเข้ามาครบทั้ง 3 ค่า
4 แสดงผลค่าเฉลี่ย โดยนำค่าที่เก็บไว้ใน sum หารด้วย 3
5. จบการทำงาน
```

จะเห็นว่าอัลกอริทึมตัวอย่างที่ 5.5 มีความยืดหยุ่นมากขึ้น จำนวนขั้นตอนก็น้อยลง จากเดิมที่มีการแก้ไขปัญหาเป็นรูปธรรม ก็เปลี่ยนมาเป็นแบบนามธรรมมากขึ้น สามารถเปลี่ยนแปลงจำนวนคะแนนสอบได้ง่ายกว่าเดิม เพียงแค่เปลี่ยนจำนวนการวนซ้ำ และในการหาค่าเฉลี่ยจาก 3 ให้เป็นค่าอื่นๆ ตามต้องการ

อัลกอริทึมรูปแบบซูโดโค้ด :

1. START
2. SET sum = 0 , scoreNumber = 1
3. REPEAT
    3.1 DISPLAY “Score#”
    3.2 INPUT score
    3.3 IF score >= 0 AND score <= 100 THEN
        3.3.1 sum = sum + score
        3.3.2 scoreNumber = scoreNumber + 1
    ELSE
        3.3.3 DISPLAY “Invalid value”
    ENDIF
    UNTIL scoreNumber > 3
4. DISPLAY sum/3
5. STOP

จากตัวอย่างที่ 5.5 ให้มีการรับค่าคะแนนสอบเข้ามา โดยใช้ขั้นตอนการทำงานที่เป็นการวนซ้ำรูปแบบ ขณะที่ ... จนกระทั่ง ซึ่งเป็นประโยคคำสั่งวนซ้ำ do … while ในภาษา C การรับค่าวนรอบจนกว่าจะมีการรับค่าครบ 3 ค่า และต้องอยู่ในช่วง 0-100 ไม่เช่นนั้นให้รับค่าใหม่จนกว่าจะถูกต้อง เมื่อได้ครบจำนวนจะออกจากการวนซ้ำ นำไปหาค่าเฉลี่ยและแสดงผลออกมาให้ทราบ

ในการใช้ขั้นตอนการวนซ้ำ จะต้องมีตัวแปรเพิ่มเข้ามาช่วยในการทำงานวนซ้ำ คือ ตัวแปรค่าคงที่ scoreCount ใช้กำหนดจำนวนการวนซ้ำทั้งหมด หรือใช้เป็นจำนวนคะแนนสอบที่ต้องการเก็บค่า และมีการนำตัวแปร scoreNumber ใช้เป็นค่าสถานการณ์วนซ้ำในรอบปัจจุบัน และเป็นลำดับของคะแนนสอบ ซึ่งตัวแปรทั้งสองถูกนำมาใช้ตรวจสอบเงื่อนไขให้มีการวนซ้ำอีกต่อไปหรือไม่

2. การเขียนผังงาน (Flowchart)

ผังงาน หรือ โฟลวชาร์ต เป็นแผนภาพที่ใช้ออกแบบและอธิบายการทำงานของโปรแกรมโดยอาศัยสัญลักษณ์รูปทรงต่าง ๆ ควบคู่ไปกับลูกศร แต่ละรูปในแผนภาพจะหมายถึงการทำงานหนึ่งขั้นตอน ส่วนลูกศรจะแทนลำดับการทำงานขั้นตอนต่าง ๆ รวมทั้งทิศทางการไหลของข้อมูลตั้งแต่เริ่มต้นจนได้ผลลัพธ์ตามต้องการ ระบบงานทุกชนิดที่ผ่านการวิเคราะห์เป็นลำดับขั้นตอนแล้ว จะสามารถเขียนเป็นผังงานได้

ประโยชน์ของผังงาน

- ช่วยอธิบายลำดับขั้นตอนการทำงานของโปรแกรม
- ทำให้ตรวจสอบข้อผิดพลาดของโปรแกรมได้ง่าย
- ทำให้ผู้อื่นสามารถศึกษาการทำงานของโปรแกรมและแก้ไขโปรแกรมได้ง่าย

การเขียนผังงานที่ดี

1. ใช้สัญลักษณ์ตามที่กำหนดไว้
2. ผังงานจะต้องมีจุดเริ่มต้น (Start) และสิ้นสุด (Stop/End/Finish)
3. ใช้หัวลูกศรแสดงทิศทางการไหลของข้อมูลจากบนลงล่างหรือซ้ายไปขวา (ยกเว้นที่ต้องทำซ้ำ)
4. ทุกแผนภาพต้องมีลูกศรแสดงทิศทางเข้า 1 เส้นและออก 1 เส้นโดยไม่มีการปล่อยจุดใดจุดหนึ่งไว้
5. เขียนคำอธิบายการทำงานในแต่ละขั้นตอนโดยใช้ข้อความที่สั้น กะทัดรัด ชัดเจนและเข้าใจได้ง่าย
6. ควรหลีกเลี่ยงโยงเส้นไปมาทำให้เกิดจุดตัดมาก เพราะจะทำให้เกิดข้อผิดพลาดง่าย ควรใช้สัญลักษณ์เชื่อมจุดต่อเนื่องแทน
7. ไม่ควรโยงเส้นเชื่อมผังงานที่อยู่ไกลมาก ๆ ควรใช้สัญลักษณ์จุดเชื่อมต่อแทน
8. ผังงานที่ดีควรมีความเป็นระเบียบเรียบร้อย และชัดเจน สามารถเข้าใจและติดตามขั้นตอนได้ง่าย
9. ผังงานควรมีการทดสอบความถูกต้องของการทำงานก่อนไปเขียนโปรแกรม

ประเภทการเขียนผังงาน

สามารถแบ่งออกเป็น 2 ประเภท คือ

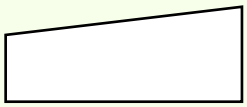
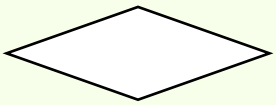
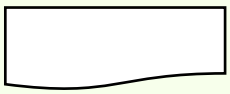
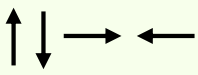
1. ผังงานระบบ (System Flowchart) ใช้แสดงขั้นตอนการทำงานในระบบงานหนึ่ง ๆ โดยกล่าวถึงข้อมูลต่าง ๆ ที่เกี่ยวข้องทั้งหมด เช่น เอกสารเบื้องต้นคืออะไร วัสดุที่ใช้คืออะไร หน่วยความจำประเภทใด

จะต้องส่งผ่านไปยังหน่วยงานใด วิธีการประมวลผลและการแสดงผลลัพธ์ โดยอาจจะกล่าวอย่างกว้าง ๆ ไม่สามารถนำมาเขียนเป็นโปรแกรมได้

2. ผังงานโปรแกรม (Program Flowchart) ผังงานประเภทนี้ จะแสดงถึงขั้นตอนของคำสั่งที่ในโปรแกรม การรับข้อมูล การประมวลผล การแสดงข้อมูล บางครั้งเรียกว่าผังการเขียนโปรแกรม

สัญลักษณ์รูปภาพของโฟลวชาร์ต

จากตารางเป็นสัญลักษณ์รูปภาพของโฟลวชาร์ตบางส่วนที่นิยมใช้งานบ่อยๆ เท่านั้น

สัญลักษณ์รูปภาพ	ความหมาย
	จุดเริ่มต้น (start) หรือจุดสิ้นสุด (stop)
	รับข้อมูล (input) หรือแสดงผลข้อมูล (output)
	รับข้อมูลนำเข้าจากคีย์บอร์ด (input from keyboard)
	รับข้อมูลนำเข้าจากคีย์บอร์ด (input from keyboard)
	การตัดสินใจ (decision) หรือการเปรียบเทียบ (compare)
	แสดงผลข้อมูลออกทางเครื่องพิมพ์ (printer)
	การทำงานย่อย (subprogram)
	จุดเชื่อมต่อ (connection)
	ทิศทาง (flow)

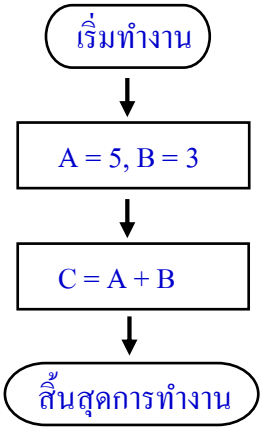
แนวคิดการเขียนผังงาน (Flowchart)

การเขียนผังงาน (โฟลวชาร์ต) มีหลักการง่ายๆ 3 ข้อ คือ

1. การทำงานแบบลำดับ (Sequence)

การทำงานแบบตามลำดับ (Sequence) เป็นรูปแบบการเขียนโปรแกรมที่ง่ายที่สุด และไม่มีความซับซ้อน การทำงานเรียงลำดับจากบนล่าง มีการทำงานที่ละคำสั่งจนจบการทำงาน ซึ่งมีรูปแบบการทำงาน

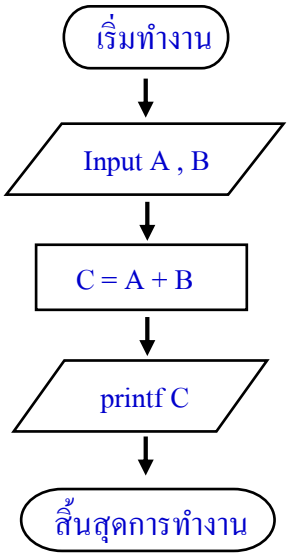
ตัวอย่าง การทำงานแบบลำดับ (Sequence)



อธิบาย :

1. เริ่มการทำงานของโปรแกรม
2. กำหนดค่าตัวแปร A = 5, B = 3
3. กำหนดให้ตัวแปร C = A + B
4. จบการทำงานของโปรแกรม

ตัวอย่าง การทำงานแบบลำดับ (Sequence)



อธิบาย :

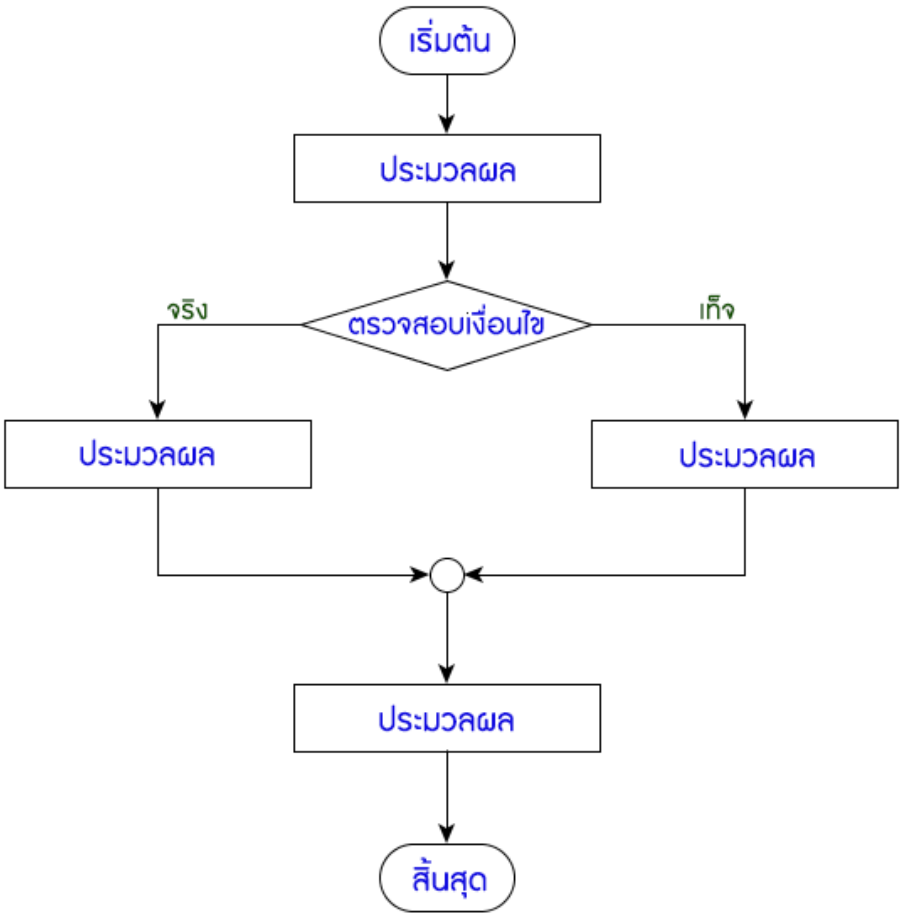
1. เริ่มการทำงานของโปรแกรม
2. รับค่าตัวแปร A , B
3. กำหนดให้ตัวแปร C = A + B
4. แสดงค่า C
5. จบการทำงานของโปรแกรม

2. การทำงานแบบทางเลือก (Decision)

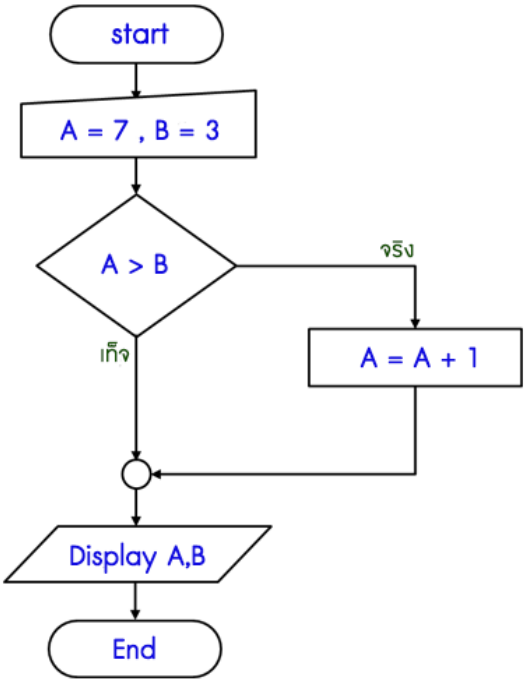
การทำงานแบบทางเลือก (Decision) เป็นรูปแบบการเขียนโปรแกรมที่มีทางเลือกเพื่อการตัดสินใจ ซึ่งโปรแกรมจะตรวจสอบเงื่อนไขเพื่อเลือกทิศทางการทำงานของโปรแกรม โดยเลือกทางเลือกใดทางเลือกหนึ่ง จากสองทางเลือก คือ

- ทางเลือกเมื่อเงื่อนไขเป็นจริง
- ทางเลือกเมื่อเงื่อนไขเป็นเท็จ

เมื่อทำงานในแต่ละทางเลือกเสร็จแล้ว โปรแกรมก็จะทำงานในขั้นตอนต่อไป



ตัวอย่างที่ การเขียนและอธิบายการทำงานของ Flowchart แบบเลือกการทำงานตามเงื่อนไข ดังนี้



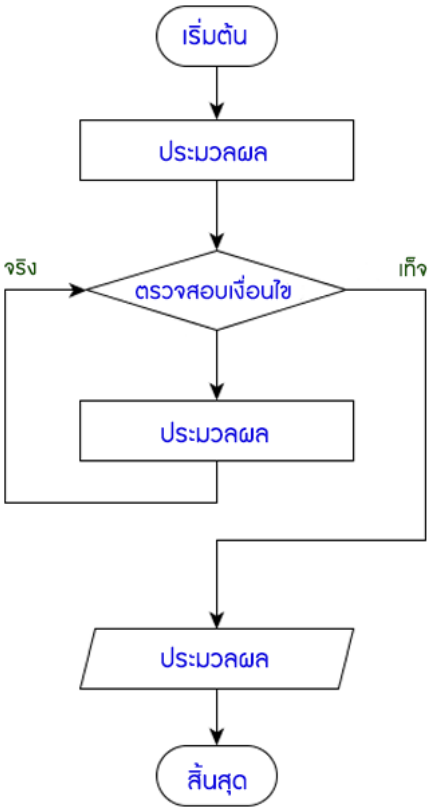
อธิบายการทำงานของ Flowchart ได้ดังนี้

1. เริ่มการทำงานของโปรแกรม โดยกำหนดค่าตัวแปร A และ B มีค่าเท่ากับ 7 และ 3
2. ตรวจสอบเงื่อนไขว่าค่าตัวแปร A มากกว่าค่าตัวแปร B หรือไม่
3. ถ้าเงื่อนไงว่าค่าตัวแปร A มากกว่าค่าตัวแปร B เป็นจริง กำหนดให้ตัวแปร A มีค่าเท่ากับ A + 1
4. แสดงค่าตัวแปร A และ B ทางจอภาพ จบการทำงานของโปรแกรม

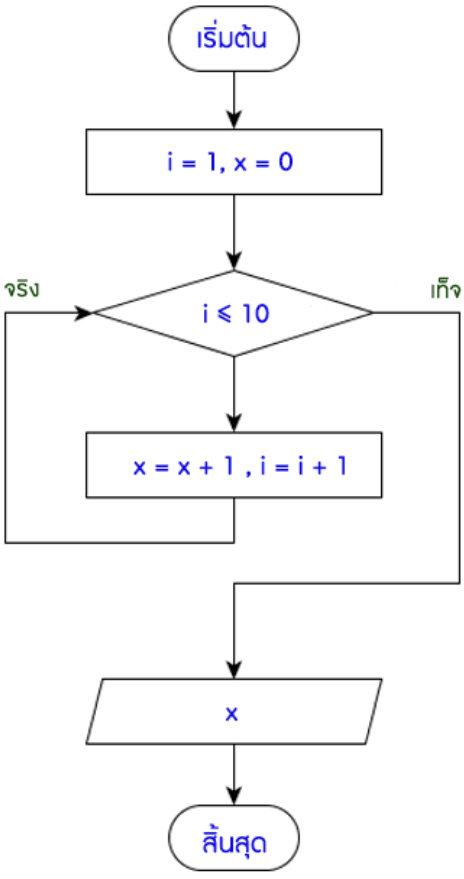
3. การทำงานแบบการทำซ้ำ (Loop)

การทำซ้ำ เป็นรูปแบบการเขียนโปรแกรมที่มีการทำงานในขั้นตอนเดิมซ้ำๆ กันหลายๆรอบ ซึ่งการทำงานของโปรแกรมจะมีการตรวจสอบเงื่อนไขเพื่อกำหนดการทำงานในลูป หรือออกจากลูปการทำงาน มีชุดคำสั่งให้ทำงานเป็นรอบอยู่ 3 รูปแบบด้วยกัน คือ

1. การทำงานเป็นรอบด้วย for
2. การทำงานเป็นรอบด้วย while
3. การทำงานเป็นรอบด้วย do-while



ตัวอย่างที่ การเขียนและอธิบายการทำงานของ Flowchart แบบการทำซ้ำ (Loop) ดังนี้



อธิบาย :

1. เริ่มการทำงานของโปรแกรม
2. กำหนดค่าตัวแปร i = 1, x = 0
3. ตรวจสอบเงื่อนไขว่า i <= 10
4. ถ้า i = 1, x = 0 จริงเพิ่มค่าให้ x = x + 1 และ i = i + 1 ถ้าเท็จให้ออกจากเงื่อนไข
5. แสดงค่า x
6. จบการทำงาน

3. การเขียนโปรแกรม (Coding)

เป็นการนำอัลกอริทึมจากการเขียนผังงาน (โฟลวชาร์ต) หรือ ซูโดโค้ด มาเขียนโปรแกรมให้ถูกต้องตามหลักไวยากรณ์ (Syntax) ของภาษา c สามารถเขียนโปรแกรมได้ดังนี้

ตัวอย่าง : จากซูโดโค้ดและโฟลวชาร์ต จงเขียนโปรแกรมรับค่าเลขจำนวนเต็ม แล้วหาผลรวม

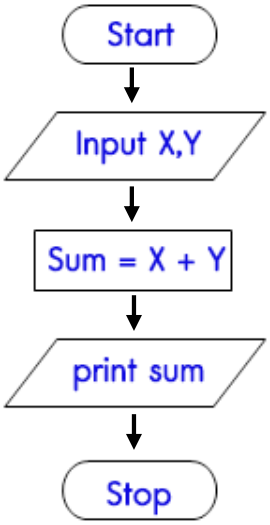
รับข้อมูลเลขจำนวนเต็ม 2 ตัวเข้ามาในโปรแกรม

กำหนดให้ x เก็บเลขจำนวนเต็มที่ 1

กำหนดให้ y เก็บเลขจำนวนเต็มที่ 2

เลขจำนวนเต็มที่ 1 + เลขจำนวนเต็มที่ 2 มีค่าเท่ากับเท่าไร

1. START
2. INPUT X
3. INPUT Y
4. COMPUTE SUM = X+Y
5. PRINT SUM
6. STOP



```c
#include <stdio.h>
main()
{
    int x,y,sum;
    printf("Value of x is : ");
    scanf("%d",&x);
    printf("Value of y is : ");
    scanf("%d",&y);
    sum = x+y;
    printf("Sum of %d + %d is %d \n",x,y,sum);
}
```

INPUT X

INPUT Y

SUM X + Y

PRINT SUM

หากนำโปรแกรม (source code) มาพิจารณา จะพบว่าการเขียนโปรแกรมมีขั้นตอนเป็นไปตามขั้นตอนอัลกอริทึมที่ได้วิเคราะห์ขึ้นทุกประการ

4. ทดสอบโปรแกรม (Testing)

หลังจากเขียนโปรแกรมจะต้องทดสอบความถูกต้องของโปรแกรมที่เขียนขึ้น หาจุดผิดพลาดของโปรแกรม และตรวจสอบจนไม่พบจุดผิดพลาดอีก จุดผิดพลาดของโปรแกรมนี้เรียกว่า บัก (bug) ส่วนการแก้ไขข้อผิดพลาดให้ถูกต้องเรียกว่า ดีบัก (debug) โดยทั่วไปแล้วข้อผิดพลาดจากการเขียนโปรแกรมจะมี 2 ประเภท คือ

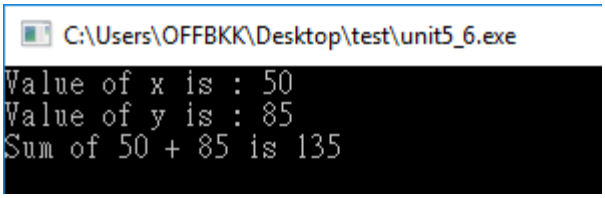
1) การเขียนคำสั่งไม่ถูกต้องตามหลักการเขียนโปรแกรมนั้นๆ ซึ่งเรียกว่า syntax error หรือ coding error ข้อผิดพลาดประเภทนี้เรามักพบตอนแปลภาษาโปรแกรมเป็นรหัสภาษาเครื่อง

2) ข้อผิดพลาดทางตรรกะ หรือ logic error เป็นข้อผิดพลาดที่โปรแกรมทางานได้ แต่ผลลัพธ์ออกมาไม่ถูกต้อง

จากโปรแกรมรับค่าเลขจำนวนเต็ม แล้วหาผลรวม สามารถทดสอบโปรแกรมได้ดังนี้

```c
#include <stdio.h>
main()
{
    int x,y,sum;
    printf("Value of x is : ");
    scanf("%d",&x);
    printf("Value of y is : ");
    scanf("%d",&y);
    sum = x+y;
    printf("Sum of %d + %d is %d \n",x,y,sum);
}
```

ผลลัพธ์ของโปรแกรม



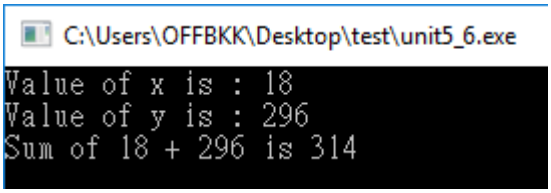
อธิบายผลลัพธ์ของโปรแกรม

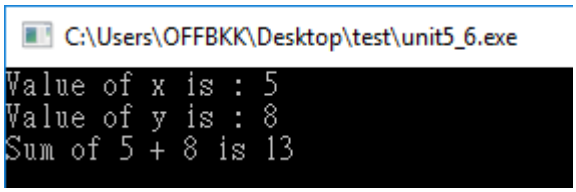
ปรากฏคำสั่ง Value of x ให้รับค่า x จากแป้นคีย์บอร์ด เสร็จแล้วกด <Enter>

ปรากฏคำสั่ง Value of y ให้รับค่า x จากแป้นคีย์บอร์ด เสร็จแล้วกด <Enter>

โปรแกรมจะนำค่าในตัวแปร x และ y ไปเก็บไว้ใน sum และแสดงผลลัพธ์ออกมาทางหน้าจอ

ทดสอบโปรแกรมโดยการทดลองรันผลลัพธ์หลายๆ ครั้ง เพื่อทดสอบความถูกต้อง





5. จัดทำคู่มือ (Documentation)

จุดประสงค์ที่สำคัญของการทำคู่มือ คือ ช่วยให้ผู้ที่ศึกษาซอร์สโค้ดของโปรแกรม (source code) เข้าใจได้ง่ายขึ้น ซึ่งเป็นประโยชน์มากสำหรับการพัฒนาโปรแกรมในอนาคต เพราะจะช่วยให้ศึกษาซอร์สโค้ดได้ง่ายและรวดเร็วขึ้น การจัดทำคู่มือผู้เขียนโปรแกรมควรจัดทำคู่มือให้มีรายละเอียดมากที่สุด

โจทย์ : จงเขียนโปรแกรมรับค่าเลขจำนวนเต็มทั้ง 2 ตัวแล้วจงหาผลบวกเลขทั้ง 2 จำนวนนั้น

จากโจทย์ตัวอย่างข้างต้น สามารถจัดทำคู่มือได้ดังนี้

ชื่อโปรแกรม : การหาผลบวกของเลขจำนวนเต็ม 2 จำนวน

ตัวแปรที่ใช้ : x เก็บค่าจำนวนเต็มตัวที่ 1

y เก็บค่าจำนวนเต็มตัวที่ 2

sum เก็บค่าผลบวกของเลขจำนวนเต็มทั้ง 2 จำนวน

ชนิดของข้อมูล : x , y , sum เป็นข้อมูลชนิดเลขจำนวนเต็ม (Integer)

วิธีแก้ปัญหา : ใช้สมการ sum = x + y

การบำรุงรักษาโปรแกรม ที่ผู้เขียนโปรแกรมจะต้องคอยตรวจสอบการใช้โปรแกรมจริง เพื่อแก้ไขข้อผิดพลาดที่อาจเกิดขึ้นในภายหลัง รวมทั้งการพัฒนาโปรแกรมให้ทันสมัยอยู่เสมอเมื่อเวลาผ่านไป

ตัวอย่างที่ 1 การเขียนโปรแกรมการหาพื้นที่สี่เหลี่ยมผืนผ้า

โจทย์ : การเขียนโปรแกรมการหาพื้นที่สี่เหลี่ยมผืนผ้า

วิเคราะห์ปัญหา :

1. ปัญหา คือ ต้องการคำนวณหาพื้นที่ของสี่เหลี่ยมผืนผ้า จากสูตร

   พื้นที่สี่เหลี่ยมผืนผ้า = กว้าง x ยาว

2. ตัวแปรที่ใช้ คือ

   Width ใช้เก็บความกว้างของรูปสี่เหลี่ยมผืนผ้า

   Long ใช้เก็บความยาวของรูปสี่เหลี่ยมผืนผ้า

   Area ใช้เก็บพื้นที่รูปสี่เหลี่ยมผืนผ้า

3. ข้อมูลนำเข้า คือ ค่าของ Width , Long

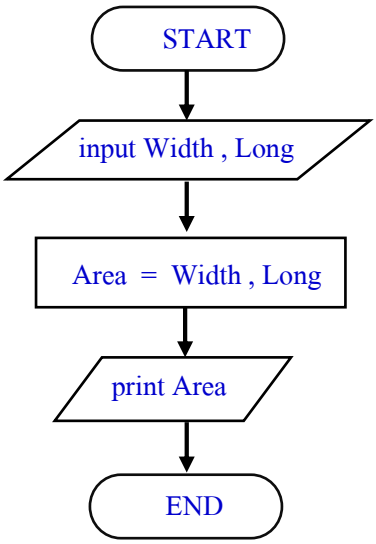
4. ผลลัพธ์ คือ พื้นที่ของสี่เหลี่ยมผืนผ้า

   Area = Width x Long

ขั้นตอนการทำงาน :

1. รับค่า Width , Long เข้ามาในโปรแกรม
2. คำนวณหาพื้นที่ของรูปสี่เหลี่ยมผืนผ้า จากสูตร Area = Width x Long
3. แสดงผลการคำนวณหาพื้นที่ของรูปสี่เหลี่ยมผืนผ้า

ขั้นตอนเขียนโฟลวชาร์ต



ขั้นตอนการเขียนโปรแกรม การคำนวณหาพื้นที่รูปสามเหลี่ยมผืนผ้า

```c
#include <stdio.h>

int main()
{
    float Width;
    float Long;

    printf("Input value of Width : ");
    scanf("%f", &Width);

    printf("Input value of Long : ");
    scanf("%f", &Long);

    printf("Area of a Rectangle is %.2f \n\n",Width * Long);
    return 0;
}
```

ผลลัพธ์ของโปรแกรม

```
C:\Users\OFFBKK\Desktop\test\unit5_4.exe
Input value of Width : 5
Input value of Long : 4
Area of a Rectangle is 20.00
```

คำอธิบายโปรแกรม

Width คือ ตัวแปรที่ใช้เก็บค่าความ กว้าง ที่เรากรอกเข้าไป

Long คือ ตัวแปรที่ใช้เก็บค่าความ ยาว ที่เรากรอกเข้าไป

Area คือ คำนวณหาพื้นที่สี่เหลี่ยมผืนผ้าตามสูตร (กว้าง x ยาว)

ตัวอย่างที่ 2 การรวบรวมคะแนนของผู้เรียน 3 คน เพื่อนำมาหาคะแนนเฉลี่ย โดยอธิบายถึงการรับค่าเข้ามา การหาคะแนนเฉลี่ย และการนำออกมาแสดงผล มีขั้นตอนดังต่อไปนี้

โจทย์ : การเขียนโปรแกรมการรับค่าคะแนนผู้เรียน 3 คน , หาคะแนนเฉลี่ย

วิเคราะห์ปัญหา :

1. เริ่มต้นทำงาน

2. กำหนดให้ sum เท่ากับ 0

3. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 1 เก็บไว้ใน score

4. นำค่าจาก score มารวมเก็บไว้ใน sum

5. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 2 เก็บไว้ใน score

6. นำค่าจาก score มารวมเก็บไว้ใน sum

7. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 3 เก็บไว้ใน score

8. นำค่าจาก score มารวมเก็บไว้ใน sum

9. แสดงผลค่าเฉลี่ย โดยนำค่าที่เก็บไว้ใน sum หารด้วย 3

10. จบการทำงาน

อัลกอริทึมรูปแบบซูโดโค้ด :

```
1. START
2. SUM = 0
3. INPUT  SCORE1
4. SUM = SUM + SCORE1
5. INPUT  SCORE2
6. SUM = SUM + SCORE2
7. INPUT  SCORE3
8. SUM = SUM + SCORE3
9. AVENGE = SUM/3
10. End
```

แสดงให้เห็นขั้นตอนการทำงานแบบเรียงลำดับที่มี 10 ขั้นตอน ตั้งแต่เริ่มต้นจนสิ้นสุดโปรแกรม ต้องทำขั้นตอนหนึ่งให้เสร็จก่อน จึงจะไปทำขั้นตอนถัดไป โดยทุกขั้นตอนจะถูกทำงานและต้องอยู่ในลำดับตำแหน่งที่เหมาะสม บางขั้นตอนสามารถสลับตำแหน่งกันได้ โดยการทำงานยังคงถูกต้อง แต่บางขั้นตอนก็ไม่สามารถสลับตำแหน่งกันได้ เพราะจะทำให้เกิดความผิดพลาดในการทำงาน

ขั้นตอนการทำงานแบบเรียงลำดับ มีลักษณะเรียบง่าย ทำความเข้าใจง่าย ตรงไปตรงมา เหมาะกับการเริ่มต้นเขียนอัลกอริทึมที่เป็นภาพรวมการทำงานทั้งหมด เพื่อทำความเข้าใจการแก้ปัญหาเป็นขั้นตอนแบบคร่าวๆ นำมาทดสอบโดยการเขียนโปรแกรมด้วยภาษา C ดังต่อไปนี้

ขั้นตอนการเขียนโปรแกรม การเขียนโปรแกรมการรับค่าคะแนนผู้เรียน 3 คน , หาคะแนนเฉลี่ย

```c
#include<stdio.h>
int main()
{
    int sum = 0,score;
    printf("score1: ");
    scanf("%d", &score);
    sum = sum + score;
    printf("score2: ");
    scanf("%d", &score);
    sum = sum + score;
    printf("score3: ");
    scanf("%d", &score);
    sum = sum + score;

    printf("Average score is %d\n",sum/3);
    return 0;
}
```

ผลลัพธ์ของโปรแกรม

```
C:\Users\OFFBKK\Desktop\test\unit5_1.exe
score1: 50
score2: 40
score3: 35
Average score is 41
```

ตัวอย่างโปรแกรม มีการรับค่าเป็นคะแนนสอบ เก็บไว้ที่ตัวแปร score และส่งค่าให้กับตัวแปร sum เพื่อรวบรวมเก็บไว้ มีการทำงานเรียงตามลำดับทั้งหมด จากนั้นทำการแสดงผลค่าเฉลี่ย ที่ได้จากการหารค่าที่เก็บไว้ในตัวแปร sum

ตัวอย่างที่ 3 การเขียนโปรแกรมคำนวณคะแนนสอบ

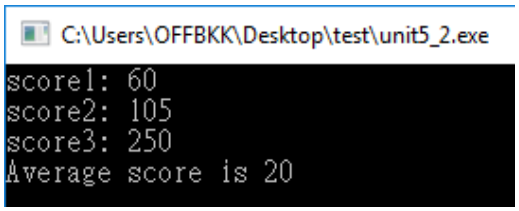
วิเคราะห์ปัญหา :

1. เริ่มต้นทำงาน
2. กำหนดให้ sum เท่ากับ 0
3. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 1 เก็บไว้ใน score
4. ถ้าค่าใน score อยู่ระหว่าง 0 -100 แล้ว
    4.1 นำไปรวมเก็บไว้ใน sum
5. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 2 เก็บไว้ใน score
6. ถ้าค่าใน score อยู่ระหว่าง 0 -100 แล้ว
    6.1 นำไปรวมเก็บไว้ใน sum
7. รับค่าคะแนนสอบ จากผู้เรียนคนที่ 3 เก็บไว้ใน score
8. ถ้าค่าใน score อยู่ระหว่าง 0 -100 แล้ว
    8.1 นำไปรวมเก็บไว้ใน sum
9. แสดงผลค่าเฉลี่ย โดยนำค่าที่เก็บไว้ใน sum หารด้วย 3
10. จบการทำงาน

การเขียนโปรแกรมตามอัลกอริทึมการทำงานแบบมีการตัดสินใจ

```c
#include<stdio.h>
int main()
{
    int sum = 0,score;
    printf("score1: ");
    scanf("%d", &score);
    if(score >= 0 && score <= 100)
        sum = sum + score;
    printf("score2: ");
    scanf("%d", &score);
    if(score >= 0 && score <= 100)
        sum = sum + score;
    printf("score3: ");
    scanf("%d", &score);
    if(score >= 0 && score <= 100)
        sum = sum + score;

    printf("Average score is %d\n",sum/3);
    return 0;
}
```

ผลลัพธ์ของโปรแกรม



โปรแกรมจะมีการตรวจสอบค่าที่รับเข้ามา โดยขั้นตอนการทำงานที่เป็นการตัดสินใจแบบ ถ้า ... แล้ว จะใช้ประโยคคำสั่งเงื่อนไข if ว่าเป็นค่าที่อยู่ในช่วง 0-100 หรือไม่ ถ้าใช่ จะนำค่าที่รับเข้ามา ส่งให้ตัวแปร sum รวมเก็บไว้ หากไม่อยู่ในช่วง ก็จะไม่มีการส่งค่าให้กับตัวแปร sum ซึ่งความหมายเท่ากับคะแนนสอบมีค่าเป็น 0 ส่งให้แทน จากนั้นนำไปหาค่าเฉลี่ยและแสดงผลออกมาให้ทราบ

หน่วยการเรียนรู้ที่ ๖ คำสั่งควบคุม

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๑ เข้าใจแนวคิดหลักของเทคโนโลยีเพื่อการดำรงชีวิตในสังคมที่มีการเปลี่ยนแปลงอย่างรวดเร็ว ใช้ความรู้และทักษะทางด้านวิทยาศาสตร์ คณิตศาสตร์ และศาสตร์อื่น ๆ เพื่อแก้ปัญหาหรือพัฒนางานอย่างมีความคิดสร้างสรรค์ด้วยกระบวนการออกแบบเชิงวิศวกรรม เลือกใช้เทคโนโลยีอย่างเหมาะสมโดยคำนึงถึงผลกระทบต่อชีวิต สังคม และสิ่งแวดล้อม

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๑ ม.๑/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์ เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็น นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนและดำเนินการแก้ปัญหา
ว ๔.๑ ม.๒/๓	ออกแบบวิธีการแก้ปัญหา โดยวิเคราะห์เปรียบเทียบ และตัดสินใจเลือกข้อมูลที่จำเป็นภายใต้เงื่อนไขและทรัพยากรที่มีอยู่ นำเสนอแนวทางการแก้ปัญหาให้ผู้อื่นเข้าใจ วางแผนขั้นตอนการทำงานและดำเนินการแก้ปัญหาอย่างเป็นขั้นตอน
ว ๔.๒ ม.๑/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงนามธรรมเพื่อแก้ปัญหาหรืออธิบายการทำงานที่พบในชีวิตจริง
ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์
ว ๔.๒ ม.๑/๓	รวบรวมข้อมูลปฐมภูมิ ประมวลผล ประเมินผล นำเสนอข้อมูล และสารสนเทศ ตามวัตถุประสงค์ โดยใช้ซอฟต์แวร์ หรือบริการบนอินเทอร์เน็ตที่หลากหลาย
ว ๔.๒ ม.๒/๒	ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้เกี่ยวกับการใช้ฟังก์ชัน if และ switch ในการตรวจสอบเงื่อนไข

๒.เรียนรู้เกี่ยวกับการใช้ฟังก์ชัน while , do while และ for ในการวนรอบการทำงานหรือทำซ้ำ

๓. เรียนรู้และฝึกทักษะจากตัวอย่างโค้ดโปรแกรม

<u>สาระการเรียนรู้</u>

- **ความรู้**

๑. ความรู้ในการเขียนโปรแกรมเกี่ยวกับการตรวจสอบเงื่อนไข

๒. ความรู้ในการเขียนโปรแกรมเกี่ยวกับการวนรอบการทำงานหรือทำซ้ำ

๓. ประยุกต์เขียนโปรแกรมได้ผลลัพธ์ที่ถูกต้อง

- **ทักษะ / กระบวนการ**

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 6.1 การควบคุมการทำงานแบบตัดสินใจ

การตรวจสอบเงื่อนไขการทำงานของโปรแกรม เป็นกระบวนการตรวจสอบเพื่อให้โปรแกรมทำงานตามขั้นตอนที่ได้กำหนดไว้ตามเงื่อนไข โดยจะมีการนำเอาค่าของตัวแปร หรือนิพจน์ต่าง มาเปรียบเทียบว่าผลลัพธ์ที่เปรียบเทียบนั้น มีค่าเป็นจริง หรือเท็จ หากมีค่าเป็นจริงก็จะทำงานตามชุดคำสั่งของเงื่อนไขนั้นๆ หากเป็นเท็จก็จะหยุดการทำงานแล้วจึงทำงานตามคำสั่งต่อไปของโปรแกรม คำสั่งที่ใช้กำหนดเงื่อนไขได้แก่ คำสั่ง if ประเภทต่างๆ และคำสั่ง Switch-case

ตัวดำเนินการเปรียบเทียบ

สัญลักษณ์	ความหมาย
>	มากกว่า
>=	มากกว่าหรือเท่ากับ
<	น้อยกว่า
<=	น้อยกว่าหรือเท่ากับ
==	เท่ากับหรือเท่ากัน
!=	ไม่เท่ากับหรือไม่เท่ากัน

ตารางแสดง ตัวดำเนินการเปรียบเทียบ

ตัวอย่าง การใช้งานตัวดำเนินการเปรียบเทียบ

```
if(a>b)
    num = a;
```

← ถ้าตัวแปร a มีค่ามากกว่าตัวแปร b ให้ตัวแปร num = a

ตัวดำเนินการลอจิก

สัญลักษณ์	ความหมาย	
&&	AND	และ
||	OR	หรือ
!	NOT	ไม่

ตัวอย่าง การใช้งานตัวดำเนินการเปรียบเทียบ

```
f(a>b && a>c)
    num = a;
```

← ถ้าตัวแปร a มีค่ามากกว่าตัวแปร b และ c ให้ตัวแปร num = a

ประโยคเงื่อนไขแบบ if

การทำงานของฟังก์ชันการตรวจสอบเงื่อนไข แบบฟังก์ชัน if ทางเลือกเดียว จะทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำงานตามประโยคคำสั่งภายในวงเล็บปีกกา แต่ถ้าเป็นเท็จจะข้ามไปทำชุดคำสั่งถัดไป ซึ่งประโยคคำสั่งภายในวงเล็บปีกกาอาจจะมีเพียงประโยคคำสั่งเดียว หรือหลายประโยคคำสั่งก็ได้ ถ้ามีเพียงประโยคคำสั่งเดียวจะไม่ใส่เครื่องหมาย ปีกกาเปิดและปิด

รูปแบบ : กรณีมีคำสั่งเดียว

```
if (expression)
        statement1;
statement2;
```

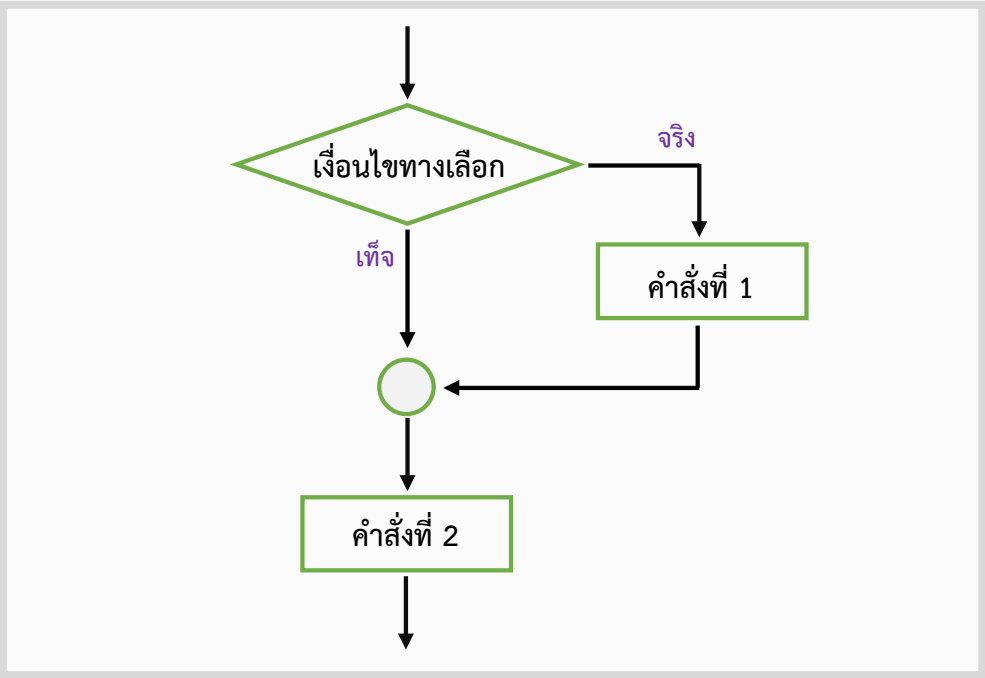
โดยที่ expression คือ นิพจน์เงื่อนไข

statement คือ ชุดคำสั่ง

รูปแบบ : กรณีมีหลายคำสั่ง

```
if (expression) {
        statement1;
        statement2;
        statement3;
}
next statement;
```

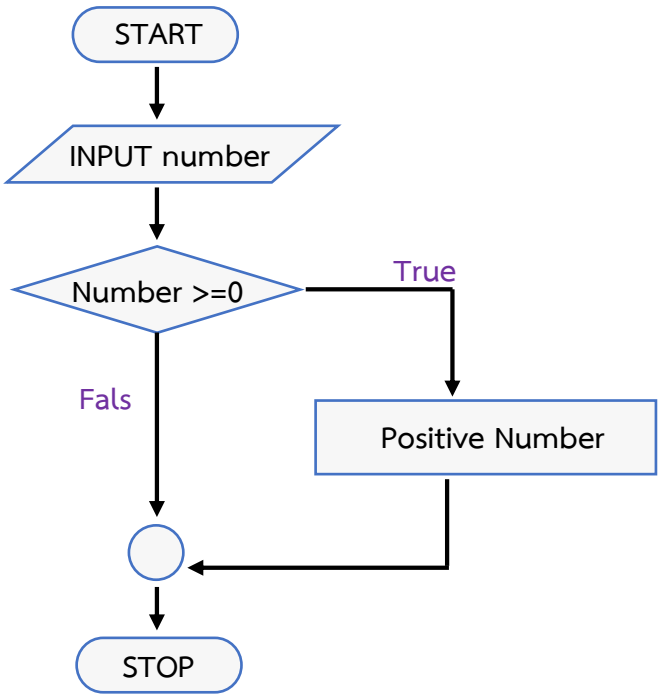
ผังงานแสดงการทำงานของคำสั่ง if :



โฟลวชาร์ต แสดงการทำงานของคำสั่งเงื่อนไข if

ตัวอย่างโปรแกรมแสดงการทำงานของคำสั่งเงื่อนไข if :

ตัวอย่างที่ 6.1 โปรแกรมตรวจสอบเลขจำนวนเต็มบวก



โฟลวชาร์ต โปรแกรมตรวจสอบเลขจำนวนเต็มบวก

โปรแกรม (source code) ตรวจสอบเลขจำนวนเต็มบวก

```
1  #include<stdio.h>
2  int main()
3  {
4      int number;
5      printf("Enter Interger Number : ");
6      scanf("%d",&number);
7      if(number>=0)
8          printf("Positive Number");
9  }
```

ผลลัพธ์การรันโปรแกรม ตรวจสอบเลขจำนวนเต็มบวก

```
C:\Users\OFFBKK\Desktop\test\unit6_1_if.exe
Enter Interger Number : 9
Positive Number
```

ปรากฏข้อความ Enter Integer Number เพื่อรับค่าตัวเลข แล้วกด <Enter> จากนั้นโปรแกรมจะเปรียบเทียบค่าตัวเลขที่ป้อนเข้ามา ถ้ามีค่ามากกว่าหรือเท่ากับ 0 จะแสดงข้อความ Positive Number

อธิบายโปรแกรม ตัวอย่างที่ 6.1 ตรวจสอบเลขจำนวนเต็มบวก

บรรทัดที่ 4 ประกาศตัวแปร number เป็นชนิดจำนวนเต็ม

บรรทัดที่ 5 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter Integer Number ออกมาทางหน้าจอ

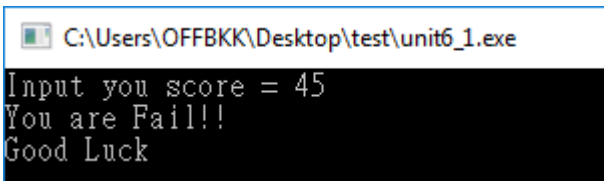
บรรทัดที่ 6 เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าตัวเลขและเก็บไว้ที่ตัวแปร number

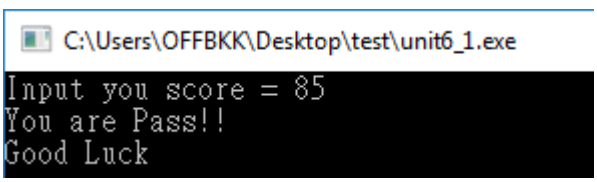
บรรทัดที่ 7-8 เรียกใช้คำสั่ง if ทำการตรวจสอบเงื่อนไข ถ้าค่าที่รับเข้ามา มากกว่าหรือเท่ากับ 0 ให้แสดงข้อความ Positive Number

ตัวอย่างที่ 6.2 โปรแกรมสำหรับการเปรียบเทียบคะแนนสอบ

```
1   #include<stdio.h>
2   main()
3   {
4       int score;
5       printf("Input you score = ");
6       scanf("%d",&score);
7       if (score>=50)
8           printf("You are Pass!!\n");
9       if (score<50)
10          printf("You are Fail!!\n");
11      printf("Good Luck");
12  }
```

ผลลัพธ์โปรแกรม เมื่อสั่งรันโปรแกรมจะปรากฏข้อความขึ้นมาให้เราป้อนคะแนน แล้วกด <Enter>





อธิบายโปรแกรม ตัวอย่างที่ 6.2

บรรทัดที่ 4 สร้างตัวแปร score เพื่อเก็บค่าของเลขจำนวนเต็ม

บรรทัดที่ 5 เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Input you score = ออกมาทางหน้าจอ

บรรทัดที่ 6 เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าคะแนนและเก็บไว้ที่ตัวแปร score

บรรทัดที่ 7-8 เรียกใช้คำสั่ง if ทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขแรกเป็นจริง คือคะแนนสอบที่รับเข้ามามากกว่าหรือเท่ากับ 50 ให้แสดงข้อความ “You are Pass!!” แต่ถ้าเงื่อนไขเป็นเท็จข้ามไปทำเงื่อนไขคำสั่งที่ 2 (บรรทัดที่ 9-10) ทันที

บรรทัดที่ 9-10 หากคะแนนสอบที่รับเข้ามาน้อยกว่า 50 ให้แสดงข้อความ “You are Fail”

ประโยคเงื่อนไขแบบ if – else

เงื่อนไข if แบบสองทางเลือก เป็นคำสั่งที่ใช้กำหนดให้โปรแกรมตัดสินใจเลือกทำคำสั่งอย่างใดอย่างหนึ่งจาก 2 ทางเลือก โดยตรวจสอบเงื่อนไขที่กำหนดว่าเป็นจริง(True) หรือเท็จ (false) ถ้าเงื่อนไขที่กำหนดเป็นจริง โปรแกรมจะทำงานที่ชุดคำสั่งที่อยู่ภายใต้คำสั่ง if แต่ถ้าเงื่อนไขที่กำหนดให้เป็นเท็จ โปรแกรมจะทำงานที่ชุดคำสั่งที่อยู่ภายใต้คำสั่ง else

รูปแบบ : กรณีมีคำสั่งเดียว

```
if (expression)
        statement1;
else
        statement2;
next statement;
```

โดยที่

expression คือ ตัวแปร ตัวดำเนินการ ค่าคงที่ นิพจน์เงื่อนไข

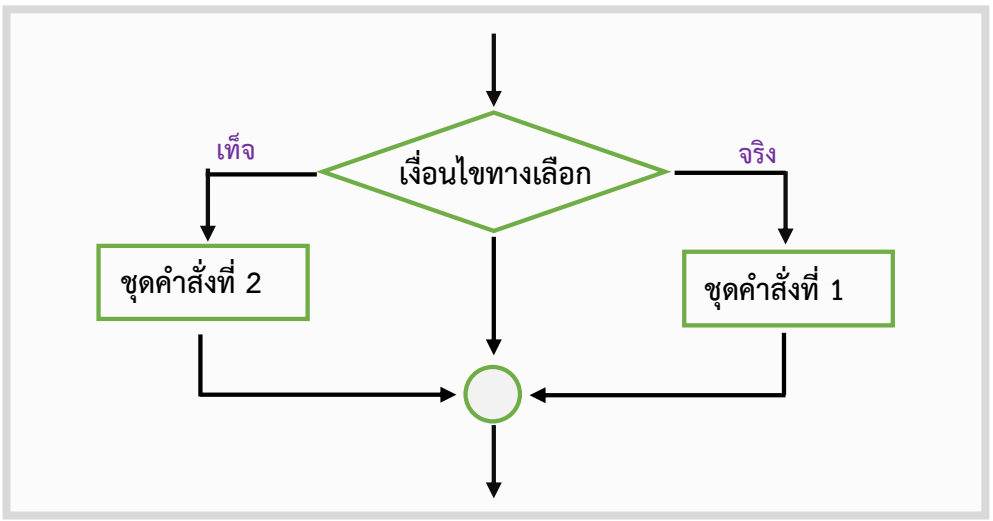
statement คือ ชุดคำสั่ง

\

รูปแบบ : กรณีมีหลายคำสั่ง

```
if (expression)
{
        statement1_1;
        statement1_2;
}
else
{
        statement2_1;
        statement2_2;
}
next statement;
```

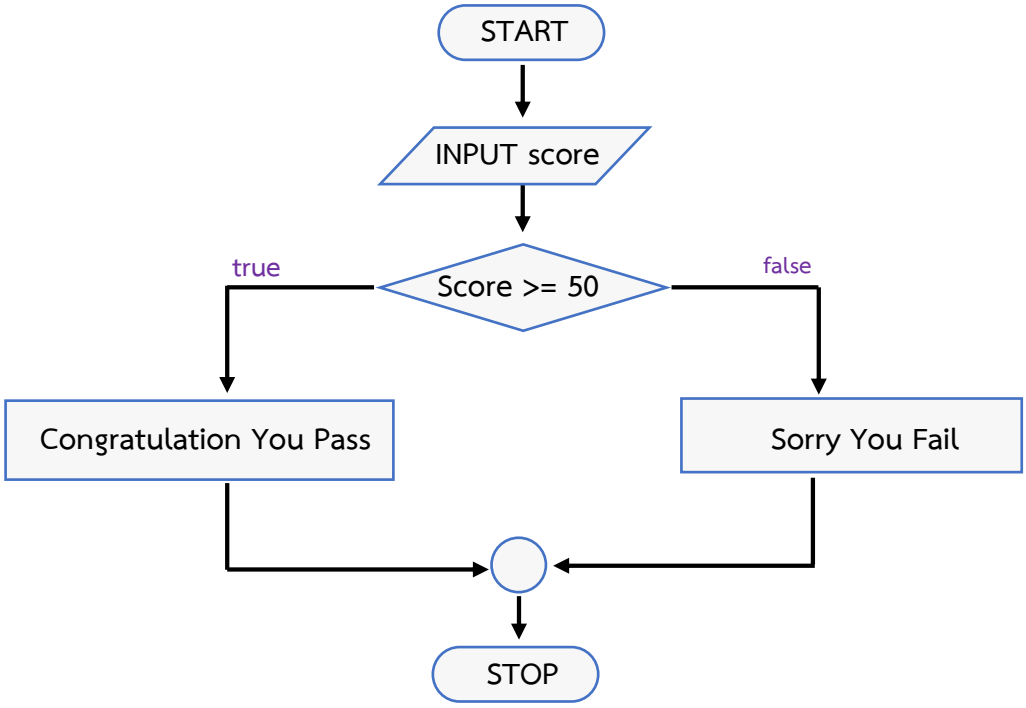
ผังงานแสดงการทำงานของคำสั่ง if-else :



โฟลวชาร์ตแสดงการทำงานของคำสั่งเงื่อนไข if-else

ตัวอย่างโปรแกรมแสดงการทำงานของคำสั่งเงื่อนไข if-else กรณีมีคำสั่งเดียว:

ตัวอย่างที่ 6.3 โปรแกรมสำหรับการเปรียบเทียบคะแนนสอบ



โฟลวชาร์ต แสดงการเปรียบเทียบคะแนนสอบ

โปรแกรมการเปรียบเทียบคะแนนสอบ

```
1   #include<stdio.h>
2   #include<conio.h>
3   main()
4   {
5       int score;
6       printf("Input you score = ");
7           scanf("%d",&score);
8       if(score>=50)
9           printf("\n--Congratulation You Pass--\n");
10      else
11          printf("\n--Sorry You Fail--\n");
12      printf("\n*Thank you so much*");
13      getch();
14  }
```

ผลลัพธ์โปรแกรม

เมื่อรันโปรแกรมเครื่องจะแสดงข้อความ Input you score ให้ป้อนค่าคะแนนเข้าไป ถ้าป้อนตัวเลขที่มีค่ามากกว่าหรือเท่ากับ 50 จะแสดงข้อความ Congratulation You Pass แต่ถ้าป้อนค่าตัวเลขน้อยกว่า 50 จะแสดงข้อความ Sorry You Fail ไม่ว่าเงื่อนไขจะเป็นจริงหรือเท็จ ก็จะแสดงข้อความ Thank you so much

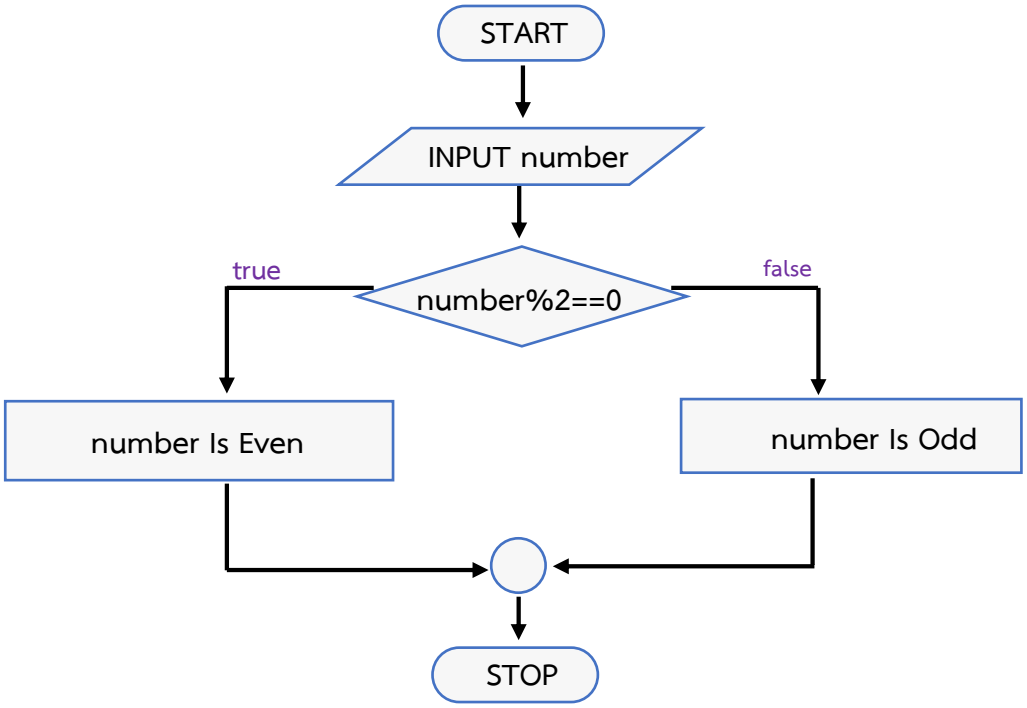
```
C:\Users\OFFBKK\Desktop\test\unit6_3.exe
Input you score = 95

--Congratulation You Pass--

*Thank you so much*
```

```
C:\Users\OFFBKK\Desktop\test\unit6_3.exe
Input you score = 20

--Sorry You Fail--

*Thank you so much*
```

อธิบายโปรแกรม ตัวอย่างที่ 6.3

บรรทัดที่ 5	สร้างตัวแปร score เพื่อเก็บค่าของเลขจำนวนเต็ม
บรรทัดที่ 6	เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Input you score = ออกมาทางหน้าจอ
บรรทัดที่ 7	เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าคะแนนและเก็บไว้ที่ตัวแปร score
บรรทัดที่ 8-9	เรียกใช้คำสั่ง if ทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขแรกเป็นจริง คือคะแนนสอบที่รับเข้ามามากกว่าหรือเท่ากับ 50 ให้แสดงข้อความ “Congratulation You Pass” แต่ถ้าเงื่อนไขเป็นเท็จข้ามไปทำเงื่อนไขในคำสั่ง else
บรรทัดที่ 10-11	เรียกใช้คำสั่ง if ทำการตรวจสอบเงื่อนไขว่า หากคะแนนสอบที่รับเข้ามาน้อยกว่า 50 ให้แสดงข้อความ “Sorry You Fail”
บรรทัดที่ 12	เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Thank you so much ออกมาทางหน้าจอ

ตัวอย่างที่ 6.4 โปรแกรมตรวจสอบเลขคู่ เลขคี่

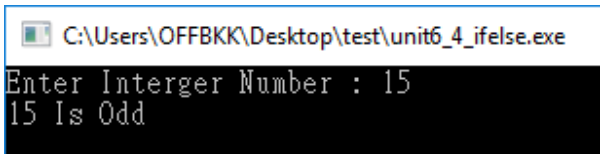


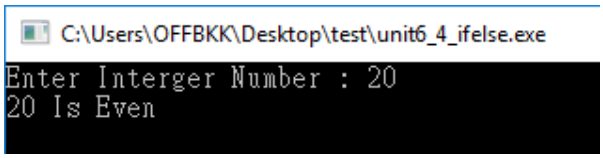
โฟลวชาร์ต แสดงการเปรียบเทียบคะแนนสอบ

โปรแกรม (source code) ตรวจสอบเลขคู่ เลขคี่

```c
#include<stdio.h>
int main()
{
    int number;
    printf("Enter Interger Number : ");
    scanf("%d",&number);

    if((number%2)==0)
    {
       printf("%d Is Even",number);
    }
    else
    {
       printf("%d Is Odd",number);
    }
    printf("\n");
}
```

ผลลัพธ์การรันโปรแกรม ตรวจสอบเลขคู่ เลขคี่





เมื่อค่าที่รับเข้ามาเป็นเลขคี่ โปรแกรมจะแสดงข้อความ Is Even แต่ถ้าเป็นเลขคู่ จะแสดงข้อความ Is Odd

อธิบายโปรแกรม ตรวจสอบเลขคู่ เลขคี่

บรรทัดที่ 4	สร้างตัวแปร number เพื่อเก็บค่าของเลขจำนวนเต็ม
บรรทัดที่ 5	เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter Integer Number ออกมาทางหน้าจอ
บรรทัดที่ 6	เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าตัวเลขและเก็บไว้ที่ตัวแปร number
บรรทัดที่ 8-11	เรียกใช้คำสั่ง if – else ทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขแรกเป็นจริง คือค่าที่รับเข้ามา หารด้วย 2 เท่ากับ 0 ให้แสดงข้อความ Is Even แต่ถ้าเงื่อนไขเป็นเท็จข้ามไปทำเงื่อนไขในคำสั่ง else
บรรทัดที่ 12-15	เงื่อนไขที่สอง คือค่าที่รับเข้ามา หารด้วย 2 เท่ากับ 0 เป็นเท็จ ให้แสดงข้อความ Is Odd

โปรแกรมแสดงการทำงานของคำสั่งเงื่อนไข if-else กรณีมีหลายคำสั่ง

ตัวอย่างที่ 6.5 โปรแกรมการคำนวณเกรด

```
1  #include <stdio.h>
2  main()
3  {
4      int score;
5      char grade;
6      printf("Input score : ");
7      scanf("%d",&score);
8      if (score >= 80)
9          grade = 'A';
10     else if (score >= 70)
11         grade = 'B';
12     else if (score >= 60)
13         grade = 'C';
14     else if (score >= 50)
15         grade = 'D';
16     else
17         grade = 'F';
18         printf("You grade is %c",grade);
19 }
```

ผลลัพธ์โปรแกรม ตัวอย่างที่ 6.5

```
C:\Users\OFFBKK\Desktop\test\unit6_4.exe
Enter score : 80
You grade is A
```

อธิบายโปรแกรม ตัวอย่างที่ 6.5

บรรทัดที่ 4 สร้างตัวแปร score เพื่อเก็บค่าของเลขจำนวนเต็ม

บรรทัดที่ 5 สร้างตัวแปร grade เพื่อเก็บค่าของเกรดที่ได้

บรรทัดที่ 7 เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าคะแนนและเก็บไว้ที่ตัวแปร score

บรรทัดที่ 8-9 เรียกใช้คำสั่ง if ทำการตรวจสอบเงื่อนไข ถ้า score >= 80 ได้เกรด A

บรรทัดที่ 10-11 มิเช่นนั้นถ้า score >= 70 ได้เกรด B

บรรทัดที่ 12-13 มิเช่นนั้นถ้า score >= 60 ได้เกรด C

บรรทัดที่ 14-15 มิเช่นนั้นถ้า score >= 50 ได้เกรด D

บรรทัดที่ 16-17 มิเช่นนั้น จะได้เกรด F

บรรทัดที่ 18 เรียกใช้ฟังก์ชัน printf() แสดงเกรดที่ได้ออกมาทางหน้าจอ

ประโยคเงื่อนไขแบบหลายทางเลือก if-else เชิงซ้อน (Nested if)

ฟังก์ชัน if หลายทางเลือกจะทำการตรวจสอบเงื่อนไขตามประโยคคำสั่งชุดที่ 1 ถ้าเงื่อนไขเป็นจริงจะทำงานตามประโยคคำสั่งชุดที่ 1 ถ้าเป็นเท็จจะทำการตรวจสอบเงื่อนไขต่อไป ตามประโยคคำสั่งชุดที่ 2 ถ้าเงื่อนไขชุดที่ 2 เป็นจริงจะทำงานตามประโยคคำสั่งชุดที่ 2 แต่ถ้าเป็นเท็จอีกก็จะตรวจสอบเงื่อนไขชุดที่ 3 ต่อไปจนถึงเงื่อนไขสุดท้าย ถ้าตรงกับเงื่อนไขใดก็จะทำงานตามประโยคคำสั่งของชุดเงื่อนไขนั้น

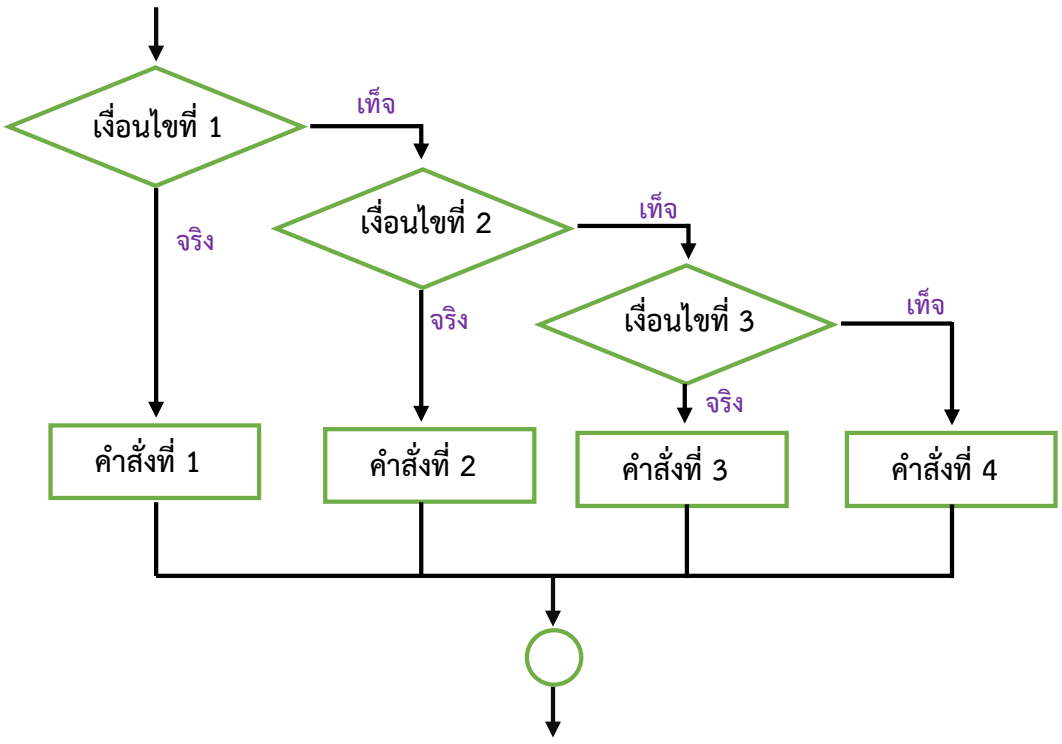
รูปแบบ :

```
if (expression1) {
statement1;
}
else if (expression2) {
statement2;
}
else if (expression3) {
statement3;
}
else {
statement4;
}
statement5;
```

โดยที่ expression คือ นิพจน์เงื่อนไข

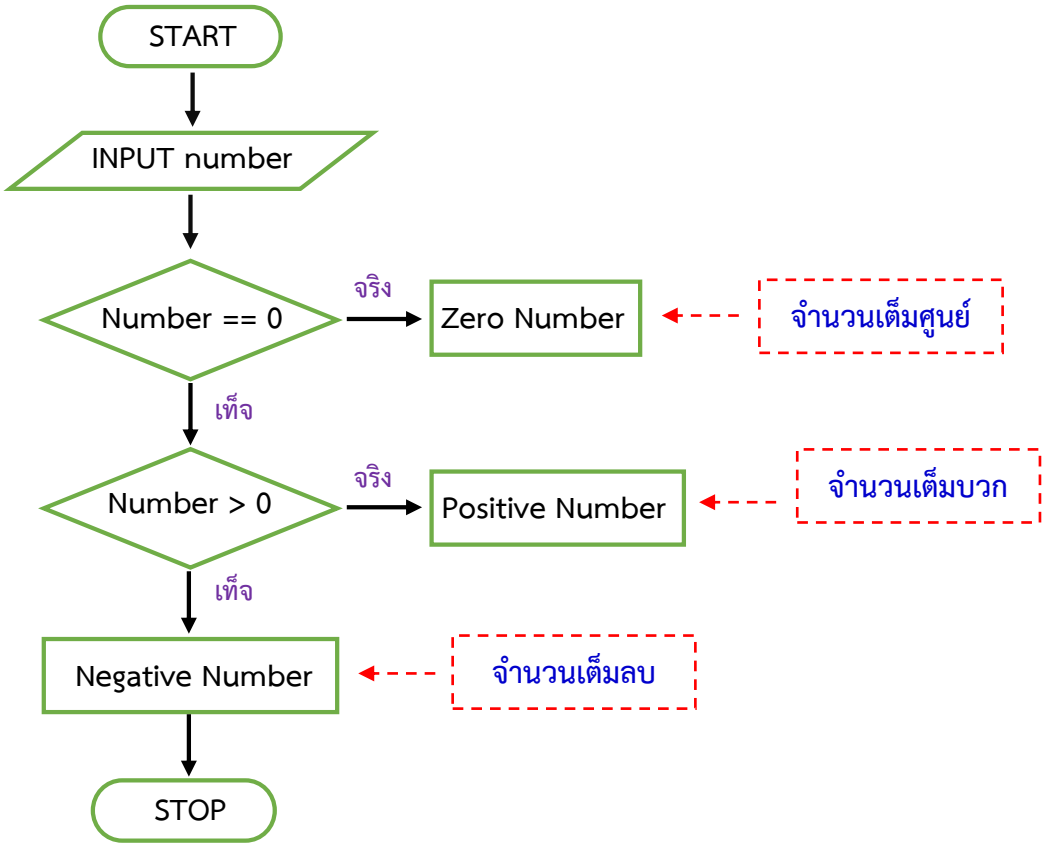
statement คือ ประโยคคำสั่ง

ผังงานแสดงการทำงานของคำสั่ง if-else :



โปรแกรมแสดงการทำงานของคำสั่งเงื่อนไข Nested if

ตัวอย่างที่ 6.6 โปรแกรมตรวจสอบเลขจำนวนเต็มบวก จำนวนเต็มลบ จำนวนเต็มศูนย์

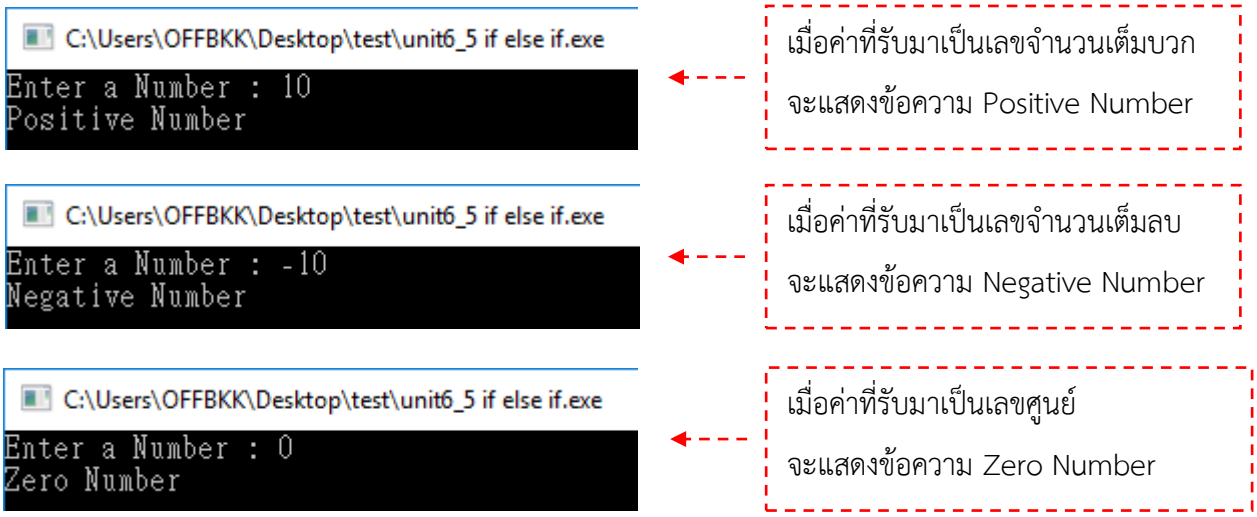


โฟลวชาร์ต แสดงการตรวจสอบเลขจำนวนเต็มบวก จำนวนเต็มลบ จำนวนเต็มศูนย์

โปรแกรม (source code) ตรวจสอบเลขจำนวนเต็มบวก จำนวนเต็มลบ จำนวนเต็มศูนย์

```
1   #include<stdio.h>
2   int main()
3   {
4       int number;
5       printf("Enter a Number : ");
6       scanf("%d",&number);
7       if(number==0)
8          printf("Zero Number");
9       else if(number>0)
10         printf("Positive Number");
11      else
12         printf("Negative Number");
13  }
```

ผลลัพธ์การรันโปรแกรม ตรวจสอบเลขจำนวนเต็มบวก จำนวนเต็มลบ จำนวนเต็มศูนย์



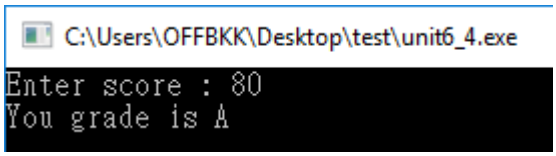
อธิบายโปรแกรม ตรวจสอบเลขจำนวนเต็มบวก จำนวนเต็มลบ จำนวนเต็มศูนย์

บรรทัดที่ 4	สร้างตัวแปร number เพื่อเก็บค่าของเลขจำนวนเต็ม
บรรทัดที่ 5	เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter a Number ออกมาทางหน้าจอ
บรรทัดที่ 6	เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าตัวเลขและเก็บไว้ที่ตัวแปร number
บรรทัดที่ 7-9	การตรวจสอบเงื่อนไข นำค่าจากตัวแปร number มาเปรียบเทียบ ถ้า number = 0 ให้แสดงข้อความ Zero Number ถ้าเป็นเท็จ ให้ไปทำที่คำสั่งถัดไป
บรรทัดที่ 10-12	การตรวจสอบเงื่อนไข นำค่าจากตัวแปร number มาเปรียบเทียบ ถ้า number > 0 ให้แสดงข้อความ Positive Number ถ้าเป็นเท็จ ให้ไปทำที่คำสั่งถัดไป
บรรทัดที่ 13-15	การตรวจสอบเงื่อนไข นำค่าจากตัวแปร number มาเปรียบเทียบ ถ้า number < 0 ให้แสดงข้อความ Negative Number ถ้าเป็นเท็จ ให้ไปทำที่คำสั่งถัดไป

ตัวอย่างที่ 6.7 โปรแกรมการคำนวณเกรด

```
1  #include <stdio.h>
2  main()
3  {
4      int score;
5      char grade;
6      printf("Input score : ");
7      scanf("%d",&score);
8      if (score >= 80)
9          grade = 'A';
10     else if (score >= 70)
11         grade = 'B';
12     else if (score >= 60)
13         grade = 'C';
14     else if (score >= 50)
15         grade = 'D';
16     else
17         grade = 'F';
18         printf("You grade is %c",grade);
19 }
```

ผลลัพธ์โปรแกรม ตัวอย่างที่ 6.7



อธิบายโปรแกรม ตัวอย่างที่ 6.7

บรรทัดที่ 4	สร้างตัวแปร score เพื่อเก็บค่าของเลขจำนวนเต็ม
บรรทัดที่ 5	สร้างตัวแปร grade เพื่อเก็บค่าของเกรดที่ได้
บรรทัดที่ 7	เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าคะแนนและเก็บไว้ที่ตัวแปร score
บรรทัดที่ 8-9	เรียกใช้คำสั่ง if ทำการตรวจสอบเงื่อนไข ถ้า score >= 80 ได้เกรด A
บรรทัดที่ 10-11	มิเช่นนั้นถ้า score >= 70 ได้เกรด B
บรรทัดที่ 12-13	มิเช่นนั้นถ้า score >= 60 ได้เกรด C
บรรทัดที่ 14-15	มิเช่นนั้นถ้า score >= 50 ได้เกรด D
บรรทัดที่ 16-17	มิเช่นนั้น จะได้เกรด F
บรรทัดที่ 18	เรียกใช้ฟังก์ชัน printf() แสดงเกรดที่ได้ออกมาทางหน้าจอ

การตรวจสอบเงื่อนไขด้วย switch-case

ฟังก์ชัน switch เป็นคำสั่งเลือกทำงานตามคำสั่ง โดยตรวจสอบค่าตัวแปรหรือนิพจน์ว่ามีค่าเท่ากับ case ใด ถ้าตรงกับ case ใดก็จะทำงานตามประโยคคำสั่งภายใต้ case นั้นทันที แต่หากตรวจสอบแล้วไม่ตรงกับ case ใดๆเลย ก็เข้าสู่การทำงานในส่วนของ default ฟังก์ชัน switch ไม่สามารถเปรียบเทียบค่ามากกว่าน้อยกว่าเหมือนฟังก์ชัน if ได้ ฟังก์ชัน switch นิยมใช้ในการตรวจสอบเงื่อนไข จำนวนหลาย ๆ เงื่อนไข

ข้อจำกัดของคำสั่ง switch-case

- ตัวแปรหรือนิพจน์ที่นำมาใช้กับฟังก์ชัน switch-case จะต้องเป็นข้อมูลชนิดเลขจำนวนเต็ม (int) ตัวอักษร (char) หรือประเภทข้อมูลอื่น ๆ ที่มีลักษณะเป็นจำนวนเต็ม เช่น short , long จะเป็น string , float , double หรือ long double ไม่ได้

- ค่าคงที่ ในแต่ละ case จะต้องเป็นข้อมูลชนิด char , short , int , long เท่านั้น

- ค่าคงที่ในแต่ละ case จะไปซ้ำกับค่าคงที่ใน case อื่น ไม่ได้

รูปแบบ :

```
switch (integer_expression)  {
        case constant_1;
                statement_1;
                break;
        case constant_2;
                statement_2;
                break;
        case constant_3;
                statement_3;
                break;
        default;
                statement_4;
 }
```

โดยที่

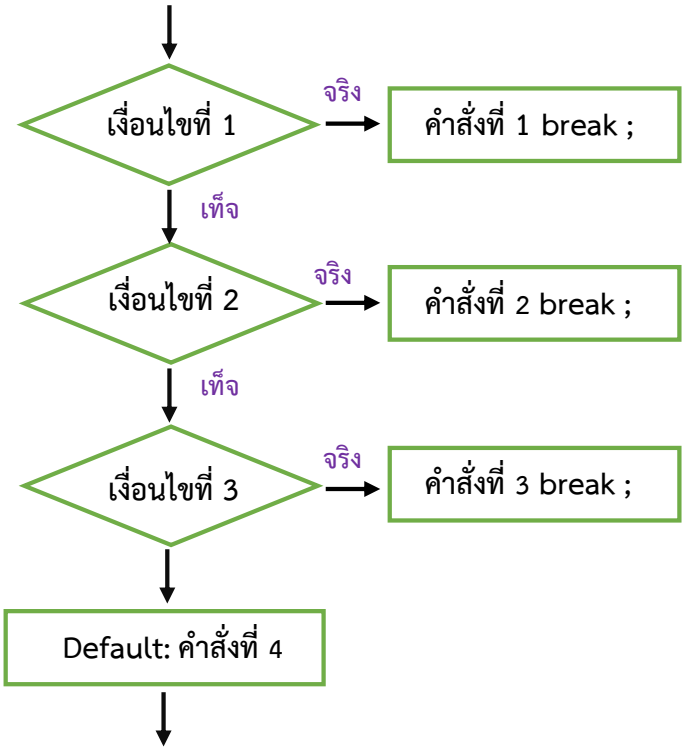
integer_expression คือ ตัวแปร/นิพจน์ที่จะตรวจสอบ

statement คือ ประโยคคำสั่ง

constant คือ ค่าคงที่

การทำงานของคำสั่ง switch-case เมื่อมีการทำงานตามคำสั่ง case ใดเสร็จสิ้นแล้ว จะต้องจบด้วยคำสั่ง break เพื่อป้องกันไม่ให้ตรวจสอบเงื่อนไขถัดไป ส่วนกรณี default นั้นไม่จำเป็นต้องมีคำสั่ง break

ผังงานแสดงขั้นตอนการทำงานของฟังก์ชัน switch-case



โฟลวชาร์ต แสดงขั้นตอนการทำงานของฟังก์ชัน switch-case

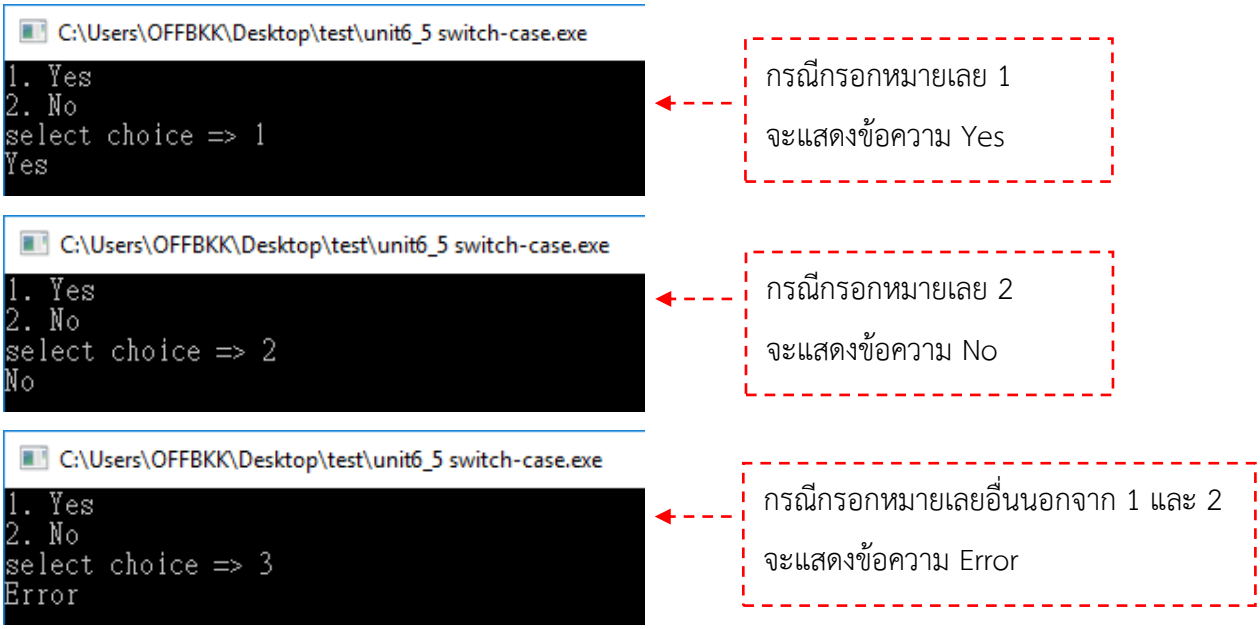
ตัวอย่างโปรแกรมการทำงานของด้วยฟังก์ชัน switch-case

ตัวอย่างที่ 6.7 โปรแกรมเลือกรายการเมนู (ตรวจสอบด้วยอักขระที่เป็นตัวเลข)

```
#include <stdio.h>
int main()
{
    int choice;
    printf("1. Yes\n");
    printf("2. No\n");
    printf("select choice => ");
    scanf("%d",&choice);
    switch (choice)
    {
        case 1 :
            printf("Yes");
            break;
        case 2 :
            printf("No");
            break;
        default :
            printf("Error");
         }
}
```

ผลลัพธ์การรันโปรแกรม

โปรแกรมจะตรวจสอบหมายเลขที่รับเข้ามา แล้วนำมาเปรียบเทียบ ถ้าตรงกับ case ใด ก็ทำตามคำสั่งที่อยู่ภายใต้ case นั้น ถ้าไม่ตรงตาม case ไหนเลย ให้ทำคำสั่ง default



คำอธิบายโปรแกรม

บรรทัดที่ 4	สร้างตัวแปร choice เพื่อเก็บค่าของเลขจำนวนเต็ม
บรรทัดที่ 5-6	เรียกใช้ฟังก์ชัน printf() แสดงข้อความที่ต้องการแสดงออกมาทางหน้าจอ ถ้าเลือก Yes ให้กรอกหมายเลข 1 ถ้าเลือก No ให้กรอกหมายเลข 2
บรรทัดที่ 7	เรียกใช้ฟังก์ชัน printf() แสดงข้อความ select choice = ออกมาทางหน้าจอ
บรรทัดที่ 8	เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าหมายเลขและเก็บไว้ที่ตัวแปร choice
บรรทัดที่ 10	ตรวจสอบค่าตัวแปร choice ที่รับมา เท่ากับค่าคงที่ใน caseใด และจะเริ่มทำคำสั่งใน case นั้น ถ้าค่า choice ไม่เท่ากับ ค่าคงที่ใน case ใดเลย จะไปเริ่มทำคำสั่งใน default
บรรทัดที่ 11-13	เรียกใช้คำสั่ง switch-case ทำการตรวจสอบเงื่อนไข ถ้าได้รับหมายเลข 1 จะแสดงข้อความ Yes ตามคำสั่ง case 1 แต่ถ้าไม่ใช่หมายเลข 1 ไปทำคำสั่ง case ถัดไป
บรรทัดที่ 14-16	เรียกใช้คำสั่ง switch-case ทำการตรวจสอบเงื่อนไข ถ้าได้รับหมายเลข 2 จะแสดงข้อความ No ตามคำสั่ง case 2 แต่ถ้าไม่ใช่หมายเลข 2 ไปทำคำสั่ง default
บรรทัดที่ 17-18	เรียกใช้คำสั่ง switch-case ทำการตรวจสอบเงื่อนไข ถ้าได้รับหมายเลขอื่นๆ นอกเหนือจากเลข 1,2 จะแสดงข้อความ Error

ตัวอย่างที่ 6.7 โปรแกรมแสดงช่วงคะแนนของเกรดที่กรอกเข้าไป

```c
#include <stdio.h>
#include <ctype.h>
int main( )
{
    char grade;
    printf("Enter your grade (A-E) : ");
    scanf("%c",&grade);
    switch(toupper(grade))
    {
        case 'A':
            printf("Your score 80-100 points\n");
            break;
        case 'B':
            printf("Your score 70-79 points\n");
            break;
        case 'C':
            printf("Your score 60-69 points\n");
            break;
        case 'D':
            printf("Your score 50-59 points\n");
            break;
        case 'E':
            printf("Your score 0-49 points\n");
            break;
        default:
            printf("Please enter character (A-E) only, Thank you");
    }
}
```

ผลลัพธ์โปรแกรม แสดงช่วงคะแนนของเกรด

เมื่อรับตัวอักษร A-E แล้ว โปรแกรมจะแสดงช่วงคะแนนของเกรดนั้น ๆ ออกมาให้ทราบ ซึ่งหากใส่ตัวอักษรอื่นๆ เข้าไปที่ไม่ใช่ตัวอักษรระหว่าง A ถึง E ก็จะแสดงข้อความ Please enter character (A-E) only, Thank you ออกมาบนหน้าจอ

```
C:\Users\OFFBKK\Desktop\test\unit6_7 switch-case.exe
Enter your grade (A-E) : A
Your score 80-100 points
```

```
C:\Users\OFFBKK\Desktop\test\unit6_7 switch-case.exe
Enter your grade (A-E) : E
Your score 0-49 points
```

```
C:\Users\OFFBKK\Desktop\test\unit6_7 switch-case.exe
Enter your grade (A-E) : F
Please enter character (A-E) only, Thank you
```

คำอธิบายโปรแกรม

บรรทัดที่ 2	ผนวกเฮดเดอร์ไฟล์เพิ่มเติมคือ #include <ctype.h>
บรรทัดที่ 5	สร้างตัวแปร char เพื่อเก็บค่าของตัวอักขระ
บรรทัดที่ 6	เรียกใช้ฟังก์ชัน printf() แสดงข้อความ Enter your grade (A-E) ออกมาทางหน้าจอ
บรรทัดที่ 7	เรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าตัวอักขระและเก็บไว้ที่ตัวแปร char
บรรทัดที่ 8	เพิ่มคำสั่ง toupper เพื่อแปลงค่าในตัวแปร char ให้เป็นตัวพิมพ์ใหญ่ จากนั้นนำค่า char ที่ได้ไปตรวจสอบเงื่อนไขตาม case ต่างๆ
บรรทัดที่ 10-12	ตรวจสอบค่าตัวแปรที่รับมา ถ้าเป็น A ให้แสดงข้อความ Your score 80-100 points
บรรทัดที่ 13-15	ตรวจสอบค่าตัวแปรที่รับมา ถ้าเป็น B ให้แสดงข้อความ Your score 70-79 points
บรรทัดที่ 16-18	ตรวจสอบค่าตัวแปรที่รับมา ถ้าเป็น C ให้แสดงข้อความ Your score 60-69 points
บรรทัดที่ 19-21	ตรวจสอบค่าตัวแปรที่รับมา ถ้าเป็น D ให้แสดงข้อความ Your score 50-59 points
บรรทัดที่ 22-24	ตรวจสอบค่าตัวแปรที่รับมา ถ้าเป็น E ให้แสดงข้อความ Your score 0-49 points
บรรทัดที่ 25-26	กรณีไม่ตรงกับ case ใดเลย ให้แสดงข้อความ Please enter character (A-E) only,

สรุปเนื้อหา

การตรวจสอบเงื่อนไขการทำงานของโปรแกรม เป็นกระบวนการตรวจสอบเพื่อให้โปรแกรมทำงานตามขั้นตอนที่ได้กำหนดไว้ตามเงื่อนไข (Condition) โดยจะมีการนำเอาค่าของตัวแปร หรือนิพจน์ต่าง ๆ มาเปรียบเทียบทางตรรกศาสตร์ ว่าผลลัพธ์ที่เปรียบเทียบนั้น มีค่าเป็นจริง หรือเท็จ หากมีค่าเป็นจริงก็จะทำงานตามคำสั่ง หรือชุดคำสั่งของเงื่อนไขนั้นๆ หากเป็นเท็จก็จะหยุดการทำงานแล้วจึงทำงาน ตามคำสั่งต่อไปของโปรแกรม คำสั่งที่ใช้กำหนดเงื่อนไขได้แก่ คำสั่ง if ประเภทต่าง ๆ และคำสั่ง Switch-case

ฟังก์ชัน if () เป็นฟังก์ชันใช้ในการเปรียบเทียบข้อมูล มีทั้งแบบทางเลือกเดียวและหลายทางเลือก

ฟังก์ชัน switch () เป็นฟังก์ชันในการเปรียบเทียบข้อมูล เหมาะกับการเปรียบเทียบข้อมูลที่มีหลายๆ ทางเลือก

ดังนั้นการศึกษาความรู้ในเรื่อง การตรวจสอบเงื่อนไข และเลือกการทำงานของโปรแกรมที่ถูกต้องจึงจะส่งผลให้นักเรียนสามารถใช้คำสั่งการตรวจสอบเงื่อนไข และคำสั่งเลือกการทำงานของโปรแกรม เพื่อนำความรู้และทักษะที่ได้ไปใช้ในการเขียนโปรแกรมด้วยภาษาซีได้อย่างถูกต้องและมีประสิทธิภาพต่อไป

บทที่ 6.2 การควบคุมการทำงานแบบวนซ้ำ (Loop)

การทำงานแบบวนรอบ หรือที่เรียกกันว่าการทำงานแบบลูป (Loop) นั้น เป็นการเขียนโปรแกรมเพื่อให้เกิดการทำงานแบบซ้ำๆ วนตามรอบที่กำหนด หรือวนรอบตามเงื่อนไขที่ได้ระบุไว้

ฟังก์ชันในการวนรอบ ในภาษาซีนั้นจะใช้คำสั่งในการวนรอบอยู่ 3 คำสั่ง ได้แก่

1. การทำซ้ำแบบที่ทราบจำนวนครั้งในการทำซ้ำ คำสั่ง for
2. การทำซ้ำแบบถ้าเงื่อนไขเป็นจริงจะทำชุดคำสั่ง คำสั่ง While
3. การทำซ้ำจนระบบมีเงื่อนไขอย่างหนึ่งจึงหยุด คำสั่ง do-while

การทำซ้ำด้วยลูป for()

ฟังก์ชัน for เป็นคำสั่งการวนรอบที่ใช้ในการเขียนโปรแกรมที่มีลักษณะการทำงานซ้ำๆ ส่วนใหญ่ใช้กับการเขียนโปรแกรมที่รู้จำนวนรอบที่แน่นอน

รูปแบบ

```
for (ค่าเริ่มต้น; เงื่อนไข; การนับจำนวนรอบ)
    {
      ชุดคำสั่งที่ 1;
      ชุดคำสั่งที่ 2;
      ชุดคำสั่งที่ 3;
      ชุดคำสั่งที่ N;
    }
```

- ค่าเริ่มต้นของตัวนับจำนวนรอบ
- เงื่อนไขของตัวแปรที่ใช้ในการกำหนดการวนรอบ
- การเพิ่ม-ลดค่าเพื่อนับการวนรอบในแต่ละรอบ

การทำงานของฟังก์ชัน for ก่อนอื่นจะต้องกำหนดค่าเริ่มต้นให้กับตัวแปร จากนั้นจึงตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริง (true) จะทำชุดคำสั่งในปีกกา{} จากนั้นทำการเพิ่มหรือลดค่าให้กับตัวแปรแล้วทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริง (true) จะทำต่อ แต่ถ้าเงื่อนไขเป็นเท็จ (false) จะออกจากลูป

รูปแบบโดยปกตินั้น เมื่อกำหนดรูปแบบคำสั่ง for แล้วคำสั่งที่จะสามารถทำงานได้จะทำงานเพียงหนึ่งคำสั่งเท่านั้น นั่นก็คือคำสั่งที่เขียนต่อจากคำสั่ง for หรือหากต้องการให้การวนรอบหนึ่งรอบสามารถทำงานได้หลายคำสั่งพร้อม ๆ กันในแต่ละรอบ สามารถที่จะใช้เครื่องหมาย { } เข้ามากำหนดส่วนของการทำงานเกี่ยวกับคำสั่งหรือ statement หลายๆ ชุดได้

ตัวอย่างการวนรอบ

```
for (i = 1; i <= 10; i++)
```

ซึ่งอธิบายได้ว่า

- กำหนดค่าเริ่มต้นของตัวนับจำนวนรอบเท่ากับ 1

- ประมวลผลในชุดคำสั่ง for จำนวน 10 รอบ

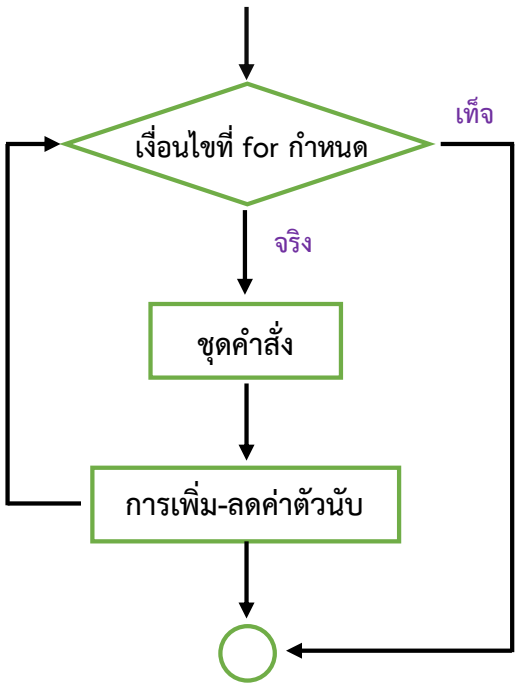
- เพิ่มค่าให้กับตัวนับจำนวนรอบทีละ 1

ตัวอย่างการวนรอบ

```
for ( i = 1; i <= 20; i+=2 )
```

แม้ว่าจะกำหนดให้จำนวนรอบทำงาน 20 รอบ แต่การเพิ่มค่าให้ตัวนับทีละ 2 จึงทำให้ลูปนี้ทำงานเพียง 10 รอบเท่านั้น

ผังงานแสดงการทำงานของคำสั่ง for :

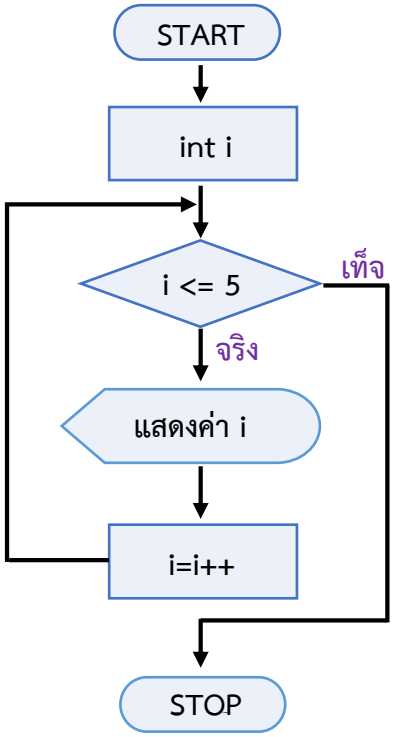


โฟลวชาร์ต แสดงการทำงานของคำสั่งวนลูป for

ตัวอย่างการเขียนโปรแกรมด้วยคำสั่งด้วยลูป for()

ตัวอย่างที่ 1 โปรแกรมพิมพ์ข้อความซ้ำๆด้วยลูป for

- ผังงาน (Flow Chart) แสดงการทำงานของโปรแกรมพิมพ์ข้อความซ้ำ



- โค๊ดโปรแกรม (Code)

```
1  #include <stdio.h>
2  main( )
3  {
4      int i;
5      for(i=1; i<=5; i++)
6          {
7              printf("%d. Coding For Kids\n",i);
8          }
9  }
```

อธิบายโปรแกรม แสดงการทำงานของโปรแกรมพิมพ์ข้อความซ้ำ

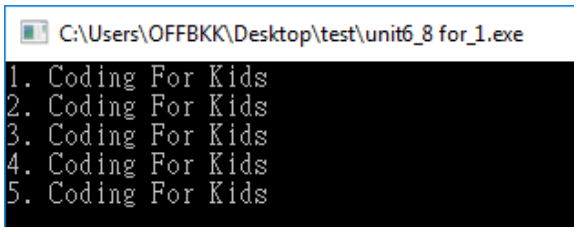
บรรทัดที่ 4 กำหนดตัวแปร i เป็นชนิดเลขจำนวนเต็ม

บรรทัดที่ 5 คำสั่ง for ทำการวนรอบ จากคำสั่ง

1. กำหนดค่าเริ่มต้นของตัวแปรเท่ากับ 1 (i = 1)
2. ตัวนับจำนวนรอบจะทำงาน 5 รอบ (i <= 5)
3. แต่ละรอบให้มีการเพิ่มจำนวนตัวนับจำนวนรอบที่ละ 1 (i ++)

บรรทัดที่ 7 ถ้าเงื่อนไขเป็นจริงให้แสดงข้อความ Coding For Kids จนกว่าจะเป็นเท็จ

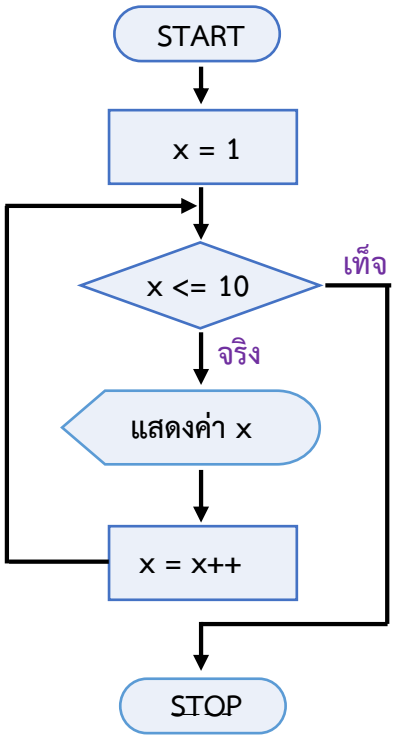
- ผลลัพธ์การรันโปรแกรม



การตรวจสอบเงื่อนไขว่า i มีค่าน้อยกว่าหรือเท่ากับ 5 หรือไม่ ถ้าตรงตามเงื่อนไขก็จะแสดงข้อความที่เก็บอยู่ในตัวแปร i แล้วทำการเพิ่มค่าขึ้นอีก 1 จากนั้นจะวนกลับขึ้นไปตรวจสอบเงื่อนไขอีกครั้ง ถ้าเงื่อนไขเป็นจริงก็จะทำงานซ้ำจนกว่าเงื่อนไขจะเป็นเท็จ จึงจะออกจากการวนรอบ

<u>ตัวอย่างที่ 2</u> โปรแกรมแสดงตัวเลขตั้งแต่ 1 – 10 ด้วยลูป for

- ผังงาน (Flow Chart) แสดงการทำงานของโปรแกรมแสดงตัวเลขตั้งแต่ 1 – 10



- โค๊ดโปรแกรม (Code)

```
1  #include <stdio.h>
2  main( )
3  {
4      int x;
5      for(x=1; x<=10; x++)
6          {
7              printf("This is round %d.\n",x);
8          }
9  }
```

- อธิบายการทำงานของโปรแกรม

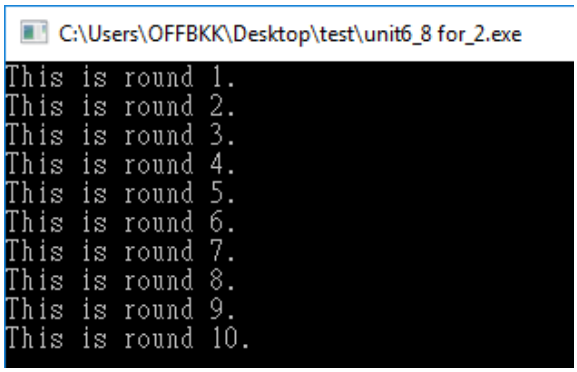
บรรทัดที่ 4 กำหนดตัวแปร x เป็นชนิดเลขจำนวนเต็ม

บรรทัดที่ 5 คำสั่ง for ทำการวนรอบ จากคำสั่ง

1. กำหนดค่าเริ่มต้นของตัวแปรเท่ากับ 1 (x = 1)
2. ตัวนับจำนวนรอบจะทำงาน 10 รอบ (x <= 10)
3. แต่ละรอบให้มีการเพิ่มค่าให้ตัวแปรจำนวนรอบที่ละ 1 (i ++)

บรรทัดที่ 7 ถ้าเงื่อนไขเป็นจริงให้แสดงข้อความ This is round 1. จนถึง This is round 10.

- ผลลัพธ์โปรแกรม



<u>ตัวอย่างที่ 3</u> โปรแกรมคำนวณแม่สูตรคูณ ตามตัวเลขที่กรอก ด้วยลูป for

```
1   #include <stdio.h>
2   int main ()
3   {
4       int i,num;
5       printf("Input multiplication number: ");
6       scanf("%d",&num);
7       for(i=1;i<=12;i++)
8       {
9          printf(" %d x %d = %d\n",num,i,i*num);
10      }
11      printf("\n********** Thank you **********");
12  }
```

โปรแกรมคำนวณแม่สูตรคูณตามแม่สูตรคูณ หรือตัวเลขที่ผู้ใช้งานกรอกเข้ามาในระบบ และจะทำการวนรอบจำนวน 12 รอบตามหลักการของแม่สูตรคูณ

- อธิบายการทำงานของโปรแกรม

บรรทัดที่ 4 กำหนดตัวแปร i , num เป็นชนิดเลขจำนวนเต็ม

บรรทัดที่ 5 คำสั่ง printf() แสดงข้อความ Input multiplication number : ออกมาทางหน้าจอ

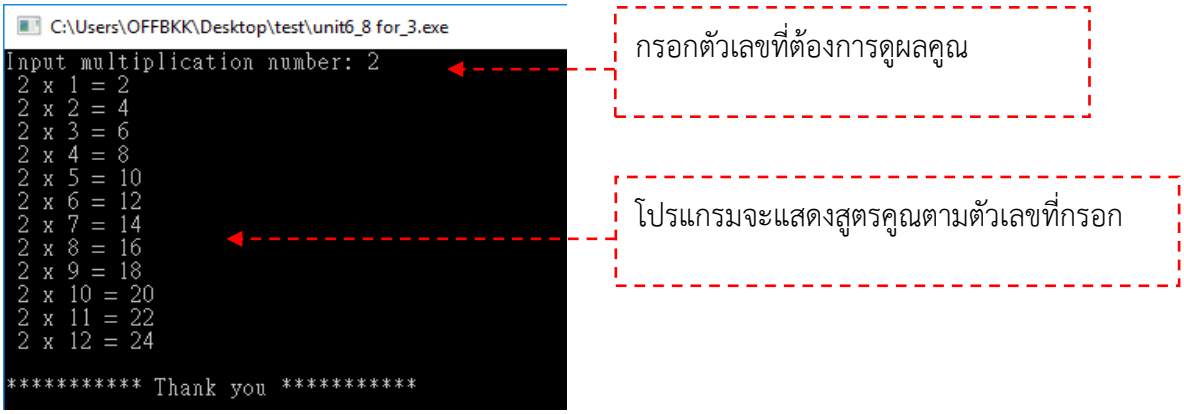
บรรทัดที่ 6 คำสั่ง scanf() เพื่อรับค่าตัวเลขสูตรคูณและเก็บไว้ที่ตัวแปร num

บรรทัดที่ 5 คำสั่ง for ทำการวนรอบ จากคำสั่ง

1. กำหนดค่าเริ่มต้นของตัวแปรเท่ากับ 1 (x = 1)
2. ตัวนับจำนวนรอบจะทำงาน 9 รอบ (x < 10)
3. แต่ละรอบให้มีการเพิ่มค่าให้ตัวแปร i จำนวนรอบที่ละ 1 (i ++)

บรรทัดที่ 9 แสดงผลลัพธ์ตัวแปร i คูณกับ num ในแต่ละรอบ แล้วขึ้นบรรทัดใหม่

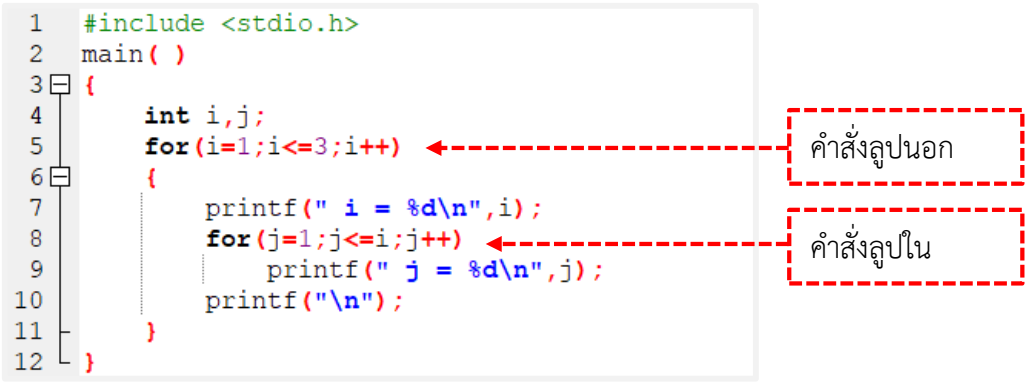
- ผลลัพธ์โปรแกรม



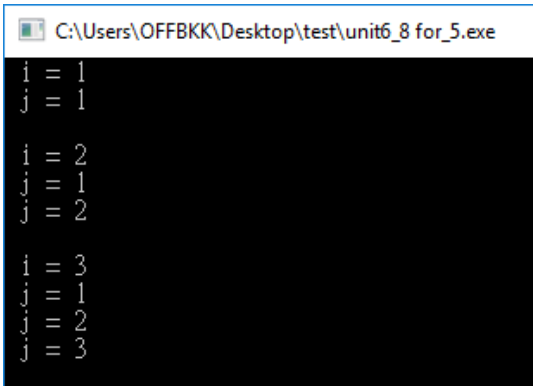
การใช้คำสั่งลูปซ้อนลูป (nested for)

เป็นการนำคำสั่งวนลูป for หลายๆชุดคำสั่งมาทำงานซ้อนกัน ซึ่งการทำงานจะเริ่มจากลูป for ที่อยู่ข้างนอกก่อน แล้วเข้าไปสู่การทำงานในลูปที่อยู่ด้านใน และทำลูปด้านในจนเสร็จก่อน ถึงออกมาทำลูปด้านนอกอีกครั้ง ซึ่งจะทำเช่นนี้ไปเรื่อยๆ จนกว่าเงื่อนไขของลูปจะเป็นเท็จ

<u>ตัวอย่างที่ 4</u> การทำงานของ nested for



- ผลลัพธ์ของโปรแกรม



- อธิบายการทำงานของโปรแกรม

จากผลลัพธ์จะเห็นว่าการทำงานในแต่ละรอบของลูป for ชั้นนอกนั้น จะต้องเข้ามาทำงานในลูปชั้นในก่อน จึงจะกลับไปทำงานลูปนอกในครั้งถัดไป ดังนั้นผลลัพธ์ที่ได้จะพบว่าแต่ละครั้งที่พิมพ์ค่า i (ลูปนอก) จะมีการพิมพ์ค่าของ j ตั้งแต่ต้นจนจบ (ลูปใน) แทรกอยู่ด้วย

บรรทัดที่ 4 กำหนดตัวแปร i , j เป็นชนิดเลขจำนวนเต็ม

บรรทัดที่ 5 ตรวจสอบเงื่อนไข i <= 3 คือ 1 <= 3 ซึ่งเป็นจริง เข้าสู่การทำงานลูปใน

บรรทัดที่ 7 คำสั่ง printf() แสดงค่าของตัวแปร i ออกมาทางหน้าจอ

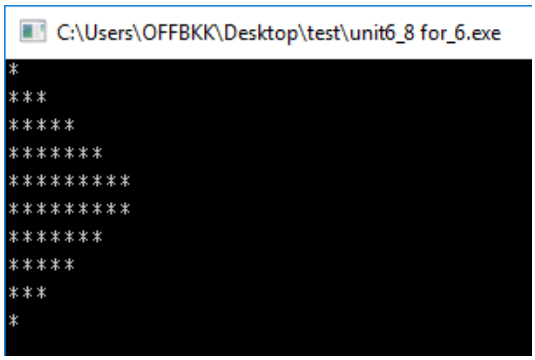
บรรทัดที่ 8 ตรวจสอบเงื่อนไข j <= 3 คือ 1 <= 3 ซึ่งเป็นจริง ถือว่าเสร็จสิ้นการทำงานลูปในวนลูปในจนครบตามเงื่อนไข จึงออกมาทำลูปนอกจนครบ

บรรทัดที่ 9 คำสั่ง printf() แสดงค่าของตัวแปร j ออกมาทางหน้าจอ

<u>ตัวอย่างที่ 5</u> โปรแกรมแสดงการพิมพ์ * ออกทางจอภาพโดยใช้คำสั่ง nested for

```
1  #include <stdio.h>
2  main( )
3  {
4      int i,j;
5      for(i=1;i<=9;i+=2)
6      {
7          for(j=1;j<=i;j++)
8              printf("*");
9          printf("\n");
10     }
11     for(i=9;i>=1;i-=2)
12     {
13         for(j=i;j>=1;j--)
14             printf("*");
15         printf("\n");
16     }
17 }
```

- ผลลัพธ์ของโปรแกรม



- อธิบายการทำงานของโปรแกรม

จากโปรแกรม มีคำสั่งวนลูปอยู่ 2 ชุด ชุดแรกคือบรรทัดที่ 7-11 และชุดที่สองคือ บรรทัดที่ 13-17 โดยคำสั่งในชุดแรก สั่งให้พิมพ์ * ออกทางจอภาพ โดยเพิ่มจำนวนการพิมพ์ขึ้นมาครั้งละ 2 คือจาก 1 ไป 3,5,7,9 ตามลำดับ คำสั่งชุดที่สอง คือจะพิมพ์ * ออกทางจอภาพ โดยลดจำนวนการพิมพ์จาก 9 ไป 7,5,3,1 ตามลำดับ

การทำงานของชุดคำสั่งแรก เริ่มจากลูปนอกก่อน รอบแรกจะมีค่า i เป็น 1 จากนั้นจะเข้าสู่การทำงานของลูปใน คือ เริ่มที่ j = 1 และจะพิมพ์ * จนกว่าเงื่อนไข j <= i จะเป็นเท็จ จึงจะกลับไปทำงานลูปนอกต่อ ซึ่งจะสิ้นสุดการทำงานของลูปนอกเมื่อเงื่อนไข i <= 9 เป็นเท็จ

การทำงานของชุดคำสั่งที่ 2 มีการทำงานเหมือนชุดแรก แต่เปลี่ยนส่วนของการกำหนดค่าเริ่มต้นให้กับโปรแกรม, เงื่อนไขที่ใช้ตรวจสอบการออกจากลูป และการปรับค่าตัวแปรเท่านั้น แต่หลักการทำงานจะเหมือนกับ

การทำซ้ำด้วยลูป while()

คำสั่ง while() เป็นคำสั่งที่ใช้ควบคุมการวนรอบของโปรแกรม โดยจะมีเงื่อนไขเป็นตัวตรวจสอบว่าหากเงื่อนไขที่กำหนดไว้เป็นจริงก็จะทำงานตามคำสั่งหรือชุดคำสั่งที่กำหนดไว้ แล้วก็จะวนไปตรวจสอบเงื่อนไขที่กำหนดไว้อีกครั้ง หากเงื่อนไขยังคงเป็นจริงก็จะทำงานในรอบต่อไปอีก จนกระทั่งเงื่อนไขเป็นเท็จจึงจะออกจากการวนรอบของ while()

รูปแบบ

```
while (เงื่อนไข)
    {
      ชุดคำสั่งที่;
    }
```

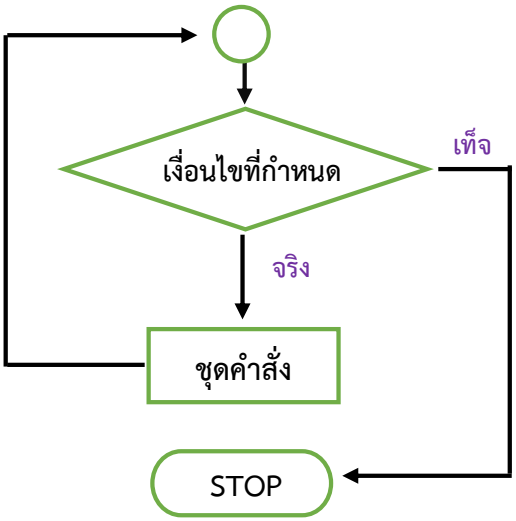
หลักการทำซ้ำด้วยลูป while()

1. ชุดคำสั่งภายในลูปจะทำงานต่อเมื่อเงื่อนไขเป็นจริง ซึ่งอาจไม่ได้รับการทำงานเลยถ้าหากเงื่อนไขนั้นเป็นเท็จตั้งแต่แรก

2. เงื่อนไขหรือนิพจน์ที่นำมาตรวจสอบ สามารถใช้ตัวดำเนินการเปรียบเทียบ หรือตัวดำเนินการทางตรรกะก็ได้ ยกตัวอย่างเช่น

```
while(i <= 10)

while(i >= 10 && j <= 20)

while(j >10 && (b < 20 || c < 30))
```

ผังงานแสดงการทำงานของคำสั่ง while :



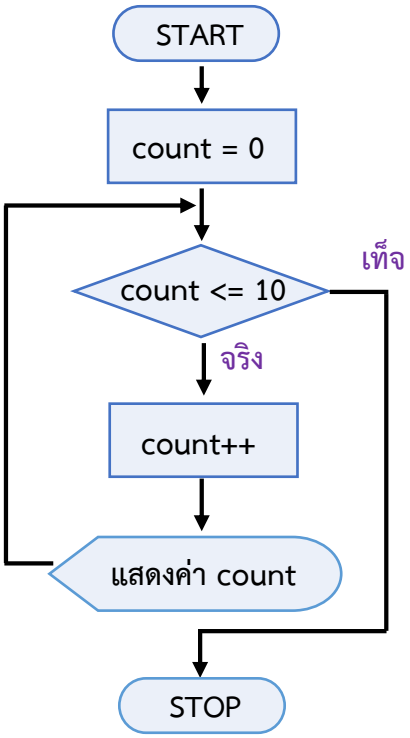
โฟลวชาร์ต แสดงการทำงานของคำสั่งวนลูป while

ตัวอย่างการเขียนโปรแกรมด้วยคำสั่งด้วยลูป while()

ตัวอย่างที่ 1 โปรแกรมวนรอบนับ 1 ถึง 10 โดยใช้ฟังก์ชัน while()

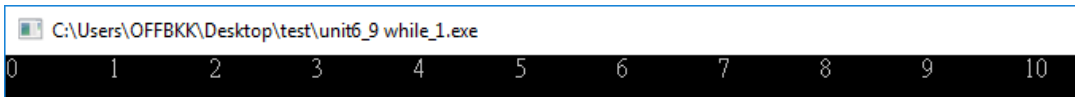
```
1  #include<stdio.h>
2  int main()
3  {
4      int count = 0;
5      while (count <= 10)
6      {
7          printf ("%d\t",count++);
8      }
9  }
```

ผังงานแสดงการทำงานของโปรแกรมวนรอบนับ 1 ถึง 10



เมื่อเริ่มต้นประมวลผล คำสั่ง while เงื่อนไขการวนซ้ำ จะถูกตรวจสอบค่า หากมีค่าเป็นจริง คำสั่งภายใต้คำสั่ง while จะถูกประมวลผล 1 รอบ แล้ววนกลับไปตรวจสอบ เงื่อนไขการวนซ้ำอีก จนกระทั่งเงื่อนไขการวนซ้ำ มีค่าเป็นเท็จ คำสั่ง while จึงสิ้นสุดลง และไปทำคำสั่งถัดไป

ผลลัพธ์การรันโปรแกรม



คำอธิบายโปรแกรม

บรรทัดที่ 4 ประกาศตัวแปร count เป็นชนิดจำนวนเต็ม มีค่าเริ่มต้น = 0

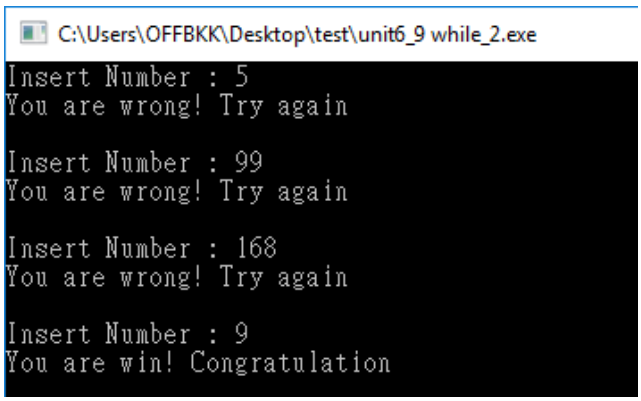
บรรทัดที่ 5 ตรวจสอบเงื่อนไข while(count<10) ถ้าเงื่อนไขเป็นจริงให้ทำชุดคำสั่งในปีกกา คือ ทำการเพิ่มค่าให้ตัวแปร count ทีละ 1 แล้ววนกลับไปตรวจสอบเงื่อนไข ทำซ้ำจนกว่าเงื่อนไขจะเป็นเท็จ

บรรทัดที่ 6 แสดงค่า count พร้อมแท็บช่องว่าง (\t)

ตัวอย่างที่ 2 โปรแกรมทายตัวเลข ด้วยคำสั่งด้วยลูป while()

```c
#include<stdio.h>
int main()
{
    int number;
    printf("Insert Number : ");
    scanf("%d",&number);
        while (number != 9)
    {
        printf("You are wrong! Try again\n");
        printf("\nInsert Number : ");
        scanf("%d",&number);
    }
    printf("You are win! Congratulation\n");
}
```

ผลลัพธ์การรันโปรแกรม

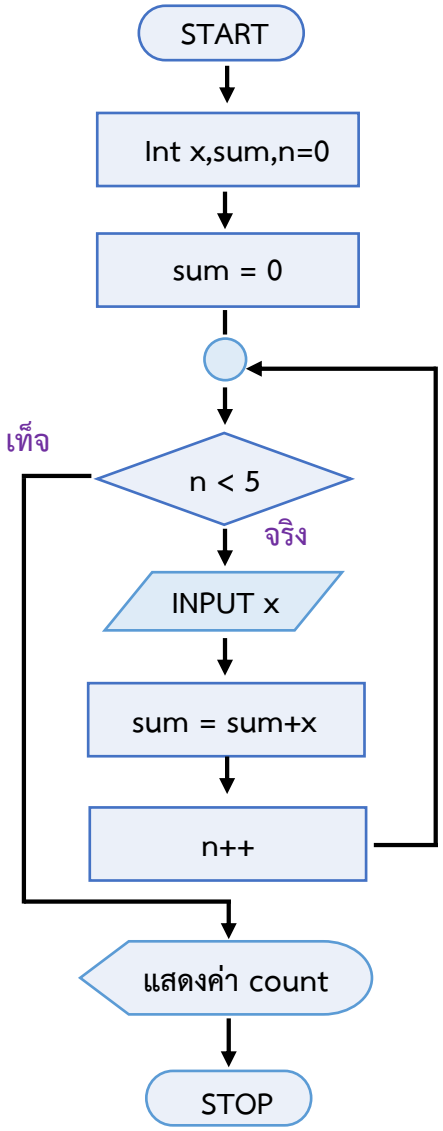


คำอธิบายโปรแกรม

เป็นโปรแกรมทายตัวเลขโดยใช้ลูป while() กำหนดการวนรอบด้วยคำสั่ง while() โดยมีเงื่อนไขว่าตัวแปร number ไม่เท่ากับ 9 (number != 9) เงื่อนไขเป็นจริง ทำงานในลูป วนจนกว่าเงื่อนไขจะเป็นเท็จนั่นคือ กรอกเลข 9 จึงจะหลุดออกจากการทำงานลูปใน

ตัวอย่างที่ 3 การเขียนโปรแกรมรับค่าตัวเลขจำนวนเต็มทีละ 1 ค่า แล้วนำตัวเลขมารวมกัน

ผังงานแสดงการทำงานของคำสั่ง โปรแกรมรับค่าตัวเลขจำนวนเต็ม แล้วนำตัวเลขมารวมกัน



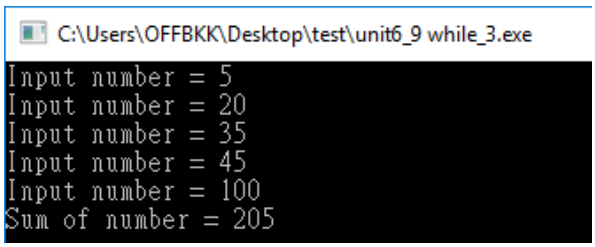
- โค๊ดโปรแกรม (source code)

```c
#include<stdio.h>
int main()
{
    int x,sum;
    int n = 0;
    sum = 0;
    while(n<5)
    {
        printf("Input number = ");
        scanf("%d",&x);
        sum=sum+x;
        n++;
    }
    printf("Sum of number = %d",sum);
}
```

อธิบายโปรแกรม

บรรทัดที่ 4-5	ประกาศตัว (x) , (sum) และ (n) เป็นจำนวนเต็มโดยกำหนดค่า n = 0
บรรทัดที่ 6	กำหนดผลรวมตัวเลข (sum) ให้มีค่าเท่ากับ 0
บรรทัดที่ 7	ตรวจสอบเงื่อนไขถ้า n < 5
	ถ้าจริง ทำงานซ้ำใน Loop
	ถ้าเท็จ แสดงผลบวก sum ทางจอภาพ
บรรทัดที่ 10	รับค่า x ทางแป้นพิมพ์
บรรทัดที่ 11	ทำตามชุดคำสั่ง sum = sum + x
บรรทัดที่ 12	เพิ่มจำนวนรอบที่ละหนึ่ง (n)

- ผลลัพธ์ของโปรแกรม



รับค่าตัวเลขจำนวนเต็มทีละ 1 ค่า จนกระทั่งรับตัวเลขครบ 5 ครั้ง แล้วนำตัวเลขมารวมกัน ให้หยุดรับค่าและแสดงผลรวมออกมาทางจอภาพ

การทำซ้ำด้วยลูป do_while()

คำสั่ง do...while เป็นคำสั่งวนซ้ำการทำงานเดิมๆ โดยโปรแกรมจะทำงานชุดคำสั่งภายในลูปก่อน 1 รอบ แล้วจึงตรวจสอบเงื่อนไขที่กำหนด ถ้าเงื่อนไขที่กำหนดเป็นจริงให้กลับไปทำงานชุดคำสั่งภายในลูปอีกครั้ง และตรวจสอบเงื่อนไขที่กำหนดอีกครั้ง ถ้าเงื่อนไขที่กำหนดเป็นเท็จโปรแกรมจะออกจากลูปการทำงานไปทำงานที่คำสั่งถัดไปทันที

รูปแบบ

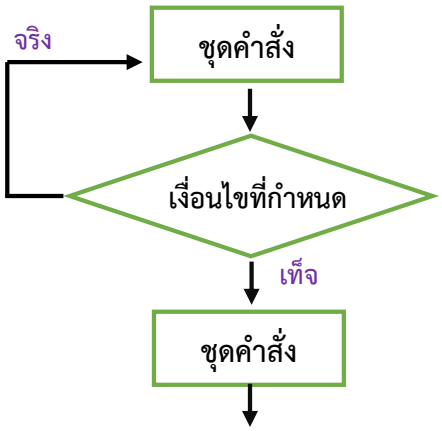
```
do
{
    คำสั่ง;
}
while (เงื่อนไข)
```

หลักการการใช้งานการวนซ้ำด้วย ลูป do_while()

1. ชุดคำสั่งภายในลูป do_while() อย่างน้อยจะต้องถูกทำงาน 1 ครั้งเสมอ แม้ว่าผลการตรวจสอบเงื่อนไขจะเป็นเท็จตั้งแต่แรกก็ตาม

2. การหลุดออกจากลูป do_while() ต่อเมื่อเงื่อนไขเป็นเท็จ

ผังงานแสดงการทำงานของคำสั่ง while :

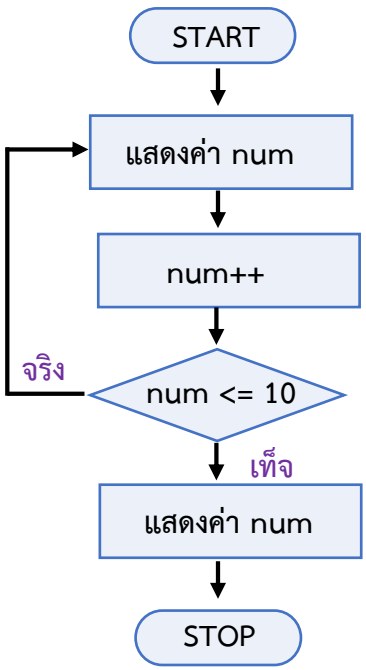


โฟลวชาร์ต แสดงการทำงานของคำสั่งวนลูป do_while

ตัวอย่างการเขียนโปรแกรมด้วยคำสั่งด้วยลูป do_while()

ตัวอย่างที่ 1 โปรแกรมนับจำนวน 1-10

- ผังงานแสดงการทำงานของคำสั่ง โปรแกรมนับจำนวน 1-10



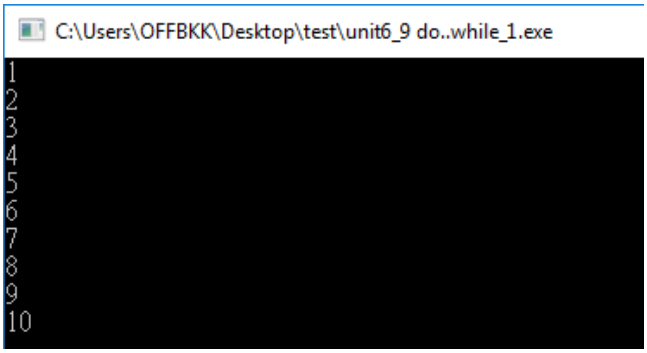
- โค๊ดโปรแกรม (source code)

```
1   #include<stdio.h>
2   main()
3   {
4       int num=1;
5       do
6       {
7           printf("%d\n",num);
8           num++;
9       }
10          while(num<=10);
11  }
```

อธิบายโปรแกรม

บรรทัดที่ 4	กำหนดตัวแปร num เป็นชนิดเลขจำนวนเต็ม มีค่า = 1
บรรทัดที่ 5	เริ่มการวนรอบด้วยคำสั่ง do
บรรทัดที่ 7	แสดงค่าของตัวแปร num(เริ่มต้นเท่ากับ 1)
บรรทัดที่ 6	กำหนดให้มีการเพิ่มค่าตัวแปรทีละ 1 (num++) สิ้นสุดการทำงานการวนรอบของคำสั่ง do
บรรทัดที่ 10	กำหนดเงื่อนไขหลังคำสั่ง while คือ num จะต้องน้อยกว่าหรือเท่ากับ 10 ถ้าผลลัพธ์ออกมาเป็นจริง ก็จะวนกลับไปทำงานตามที่คำสั่ง do กำหนด แต่หากเป็นเท็จก็จะออกจากการทำงาน

ผลลัพธ์ของโปรแกรม



โปรแกรมจะแสดงค่าตัวเลขที่เก็บอยู่ในตัวแปร num และเพิ่มค่าของตัวแปร count ขึ้นทีละ 1 หลังจากนั้นจะทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะวนกลับขึ้นไปทำงานซ้ำ จนกว่าเงื่อนไขจะเป็นเท็จ คือ num มากกกว่า 10 จึงจะจบการทำงานในลูป

หน่วยการเรียนรู้ที่ ๗ พอยน์เตอร์และอาร์เรย์

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๒ ม.๑/๒ ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์

ว ๔.๒ ม.๒/๒ ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความเข้าใจเกี่ยวกับอาร์เรย์และพอยเตอร์

๒. เรียนรู้และความเข้าใจเกี่ยวกับตัวแปรชุด (อาร์เรย์)

๓. เขียนโปรแกรมภาษาซีโดยการประยุกต์ใช้ตัวแปรชุดได้

๔. เรียนรู้ความหมายและประเภทของพอยน์เตอร์

๕. เรียนรู้การประกาศตัวแปรพอยน์เตอร์

<u>สาระการเรียนรู้</u>

- ความรู้

๑. ความรู้เกี่ยวกับการประยุกต์ใช้ตัวแปรชุด

๒. ความรู้เกี่ยวกับการประยุกต์ใช้พอยเตอร์

- ทักษะ / กระบวนการ

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 7 พอยน์เตอร์และอาร์เรย์

ความรู้พื้นฐานเกี่ยวกับพอยน์เตอร์

คุณสมบัติอันโดดเด่นของภาษา C หากไม่กล่าวถึงคงมิได้นั่นคือพอยน์เตอร์ (Pointer) หรือตัวชี้ซึ่งจัดเป็นตัวแปรชนิดหนึ่ง แต่เป็นตัวแปรชนิดพิเศษ แทนที่จะเก็บค่าข้อมูลโดยตรงเหมือนกับตัวแปรทั่วๆ ไป แต่กลับเก็บแอดเดรสที่อยู่ที่ใช้จัดเก็บค่าเหล่านั้นแทน

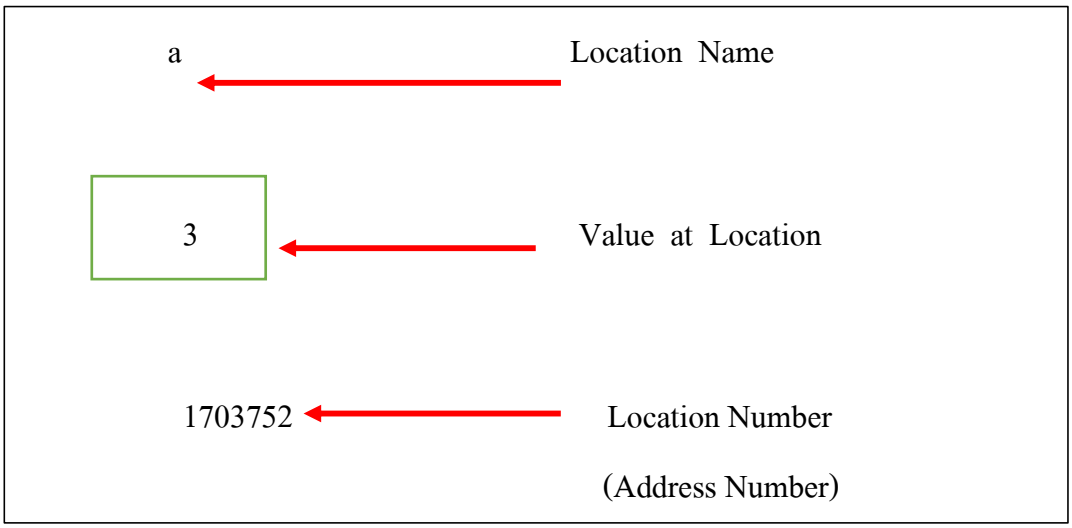
เครื่องหมายที่ใช้งานในพอยน์เตอร์

พิจารณาคำสั่งประกาศตัวแปรต่อไปนี้

```
int a = 3;
```

การประกาศตัวแปรดังกล่าวเป็นการแจ้งคอมไพเลอร์ว่า

1. ให้จองเนื้อที่ในหน่วยความจำ เพื่อจัดเก็บเลขจำนวนเต็ม
2. นำชื่อตัวแปร a เข้าไปมีส่วนร่วมกับตำแหน่งที่อยู่ของหน่วยความจำ
3. นำค่า 3 ไปเก็บไว้ในตำแหน่งนั้น



ภาพแสดงค่าที่จัดเก็บภายในหน่วยความจำ ที่อ้างอิงจากแอดเดรสและชื่อตัวแปร

จากรูปจะพบว่าคอมพิวเตอร์ได้เลือกตำแหน่งความจำหมายเลข 1703752 เพื่อจัดเก็บค่า 3 แต่หมายเลข 1703752 นั่นมิใช่ค่าของตัวแปร a แต่อย่างใด เพราะว่าในบางครั้ง คอมพิวเตอร์อาจเลือกตำแหน่งอื่นภายในหน่วยความจำเพื่อจัดเก็บค่า 3 ก็ได้ ความสำคัญที่แอดเดรสนี้จะใช้เป็นที่อยู่เพื่อจัดเก็บค่าของตัวแปร a และเราสามารถอ้างอิงค่าแอดเดรสนี้ได้ด้วยพอยน์เตอร์

ตัวอย่าง 7.1 โปรแกรมพิมพ์แอดเดรสที่ใช้เก็บข้อมูล ด้วยโอเปอเรเตอร์ &

```
1   #include <stdio.h>
2   main()
3   {
4       int a = 3;
5       printf("Address of a = %d (DEC) \n" , &a) ;
6       printf("Address of a = %p (HEX) \n" , &a) ;
7       printf("Value of a = %d \n \n" , a) ;
8   }
```

คำอธิบายโปรแกรม

บรรทัดที่ 4 ประกาศตัวแปร a ให้มีชนิดข้อมูลเป็น int มีค่าเริ่มต้นเท่ากับ 3

บรรทัดที่ 5 พิมพ์แอดเดรสที่ใช้เก็บค่า a ด้วยเลขฐานสิบ (ให้สังเกตเครื่องหมาย &)

บรรทัดที่ 6 พิมพ์แอดเดรสที่ใช้เก็บค่า a ด้วยเลขฐานสิบหก

บรรทัดที่ 7 พิมพ์ค่า a

ผลลัพธ์ของโปรแกรม

```
C:\Users\OFFBKK\Desktop\test\unit7_ex1.exe
Address of a = 6487628 (DEC)
Address of a = 000000000062FE4C (HEX)
Value of a = 3
```

สิ่งที่ต้องพิจารณาเป็นพิเศษก็คือ ในบรรทัดที่ 9-10 ได้สั่งพิมพ์แอดเดรสที่ใช้จัดเก็บค่า a โดยเครื่องหมาย & หมายความว่า “Address of” (ที่อยู่ของ) ดังนั้นนิพจน์ &a จึงคืนค่าแอดเดรสของ a ซึ่งก็คือ 1703752 (เลขฐานสิบ) หรือ 0019FF48 (เลขฐานสิบหก)

นอกจากนี้ยังมีเครื่องหมาย * ซึ่งหมายความว่า “Value at Address” (ค่าที่เก็บในแอดเดรส) ซึ่งจะชี้ไปยังแอดเดรสที่จัดเก็บค่าของตัวแปร โดยเราสามารถเรียกได้อีกชื่อ “Indirection Operator”

ตัวอย่าง 7.2 โปรแกรมพิมพ์ค่าข้อมูลที่เก็บอยู่ในแอดเดรส ด้วยโอเปอเรเตอร์ (Value at Address)

```
1   #include <stdio.h>
2    main()
3   {
4       int a = 3;
5       printf("Address of a = %u  \n" , &a);
6       printf("Address of a = %d n" , a) ;
7       printf("Value of a = %d \n \n" , *(&a) );
8   }
```

คำอธิบายโปรแกรม

บรรทัดที่ 4 ประกาศตัวแปร a ให้มีชนิดข้อมูลเป็น int มีค่าเริ่มต้นเท่ากับ 3

บรรทัดที่ 5 พิมพ์แอดเดรสของ a

บรรทัดที่ 6 พิมพ์ค่าของ a

บรรทัดที่ 7 พิมพ์ค่า a ผ่านนิพจน์ *(&a) ซึ่งหมายความว่าให้ชี้ไปยังแอดเดรสของ a แล้วนำค่าที่บรรจุอยู่ในนั้นออกมาซึ่งผลที่ได้จะเหมือนกับการสั่งพิมพ์ค่าของ a ในบรรทัดที่ 10

หน้าจอแสดงผลลัพธ์

```
C:\Users\OFFBKK\Desktop\test\unit7_ex2.exe
Address of a = 6487628
Address of a = 3 nValue of a = 3
```

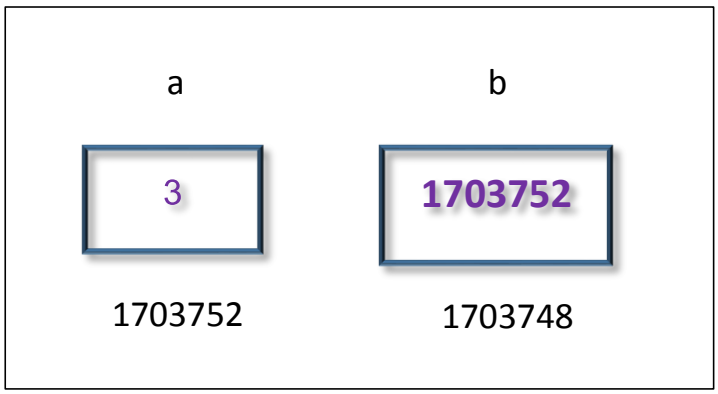
หมายเหตุ

ทั้งโอเปอเรเตอร์ & และ * ที่นำมาใช้กับตัวแปรแบบพอยน์เตอร์นั้น มิได้มีส่วนเกี่ยวข้องใด ๆ กับโอเปอเรเตอร์ทางคณิตศาสตร์ & (AND) หรือ * (การคูณ)

จากตัวอย่างโปรแกรม เมื่อสามารถใช้คำสั่ง &a เพื่อแสดงแอดเดรสของตัวแปร a แล้วเรายังสามารถนำแอดเดรสนี้ไปเก็บไว้ในตัวแปรได้เช่นกัน ดังตัวอย่างเช่น

```
b = &a;
```

ผลที่ได้จากคำสั่งดังกล่าวก็คือ ค่าที่เก็บไว้ในตัวแปร b จะเก็บแอดเดรสของ a และเมื่อ b เป็นตัวแปรคอมไพเลอร์ก็ต้องมีการจัดเตรียมพื้นภายในหน่วยความจำให้ ซึ่งเป็นไปดังรูป



ภาพแสดงแอดเดรสที่ใช้จัดเก็บค่าตัวแปรของ a และ b

จากรูป ตัวแปร a จะเก็บค่า 3 ส่วนตัวแปร b จะเก็บแอดเดรสของ a แต่อย่างไรก็ตาม เราไม่สามารถกำหนดให้ b=&a ได้โดยตรง เนื่องจาก b ยังมิได้ถูกประกาศใช้งานและด้วยตัวแปร b ถูกนำมาใช้เพื่อเก็บแอดเดรส ดังนั้น b จึงต้องถูกประกาศชนิดข้อมูลเป็นพอยน์เตอร์ดังนี้

```
int * b ;
```

นั่นหมายความว่า ตัวแปร b ก็คือตัวแปรที่ใช้ระบุ value of address นั่นเอง และเพื่อให้เกิดความเข้าใจมากยิ่งขึ้นให้พิจารณาจากโปรแกรมต่อไปนี้

ตัวอย่าง 7.3 ตัวอย่างการกำหนดตัวแปรชนิดพอยน์เตอร์ เพื่อจัดเก็บค่าแอดเดรส

```
1  #include <stdio.h>
2  main()
3  {
4      int a = 3;
5      int *b;
6      b = &a;
7      printf("Address of a = %u  \n" , &a) ;
8      printf("Address of a = %u  \n" , b) ;
9      printf("Address of b = %u  \n" , &b) ;
10     printf("Value of b = %u  \n" , b) ;
11     printf("Value of a = %d  \n" , a) ;
12     printf("Value of a = %d  \n" , * (&a) ) ;
13     printf("Value of a = %d  \n" , b) ;
14 }
```

คำอธิบายโปรแกรม

บรรทัดที่ 4 ประกาศตัวแปร a ให้มีชนิดข้อมูลเป็น int มีค่าเริ่มต้นเท่ากับ 3

บรรทัดที่ 5 ประกาศให้ b เป็นตัวแปรพอยน์เตอร์ มีชนิดข้อมูลเป็น int

บรรทัดที่ 6 กำหนดให้ b เก็บแอดเดรสของ a

บรรทัดที่ 7 พิมพ์แอดเดรสของ a

บรรทัดที่ 8 พิมพ์ค่า b

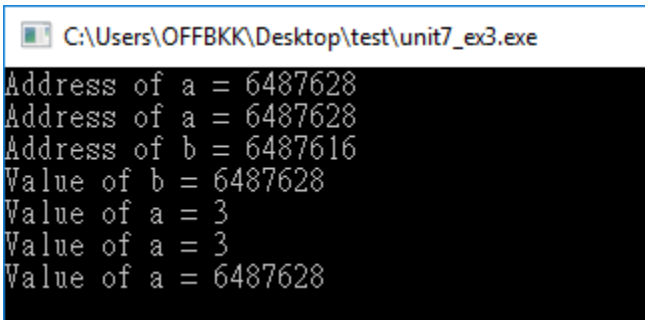
บรรทัดที่ 9 พิมพ์แอดเดรสของ b

บรรทัดที่ 10 พิมพ์ค่า b

บรรทัดที่ 11 พิมพ์ค่า a

บรรทัดที่ 12 พิมพ์ค่า a ด้วยนิพจน์ *(&a)

หน้าจอแสดงผลลัพธ์

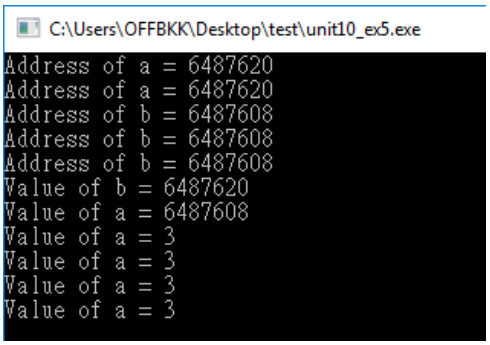


หลักการทำงานของพอยน์เตอร์ ยังสามารถขยายการทำงานเพิ่มเติมได้อีก ซึ่งเรารับรู้ในเบื้องต้นแล้วว่าพอยน์เตอร์ก็คือตัวแปรที่ใช้จัดเก็บแอดเดรส ดังนั้น พอยน์เตอร์จึงสามารถจัดเก็บที่อยู่ของพอยน์เตอร์อื่นๆ ได้เช่นกัน ด้วยการกำหนดให้พอยน์เตอร์หนึ่งชี้ไปยังพอยน์เตอร์อีกตัวหนึ่ง ที่เรียกว่า “พอยน์เตอร์ไปยังพอยน์เตอร์ (Pointers to Pointers) ” โดยใช้เครื่องหมาย **

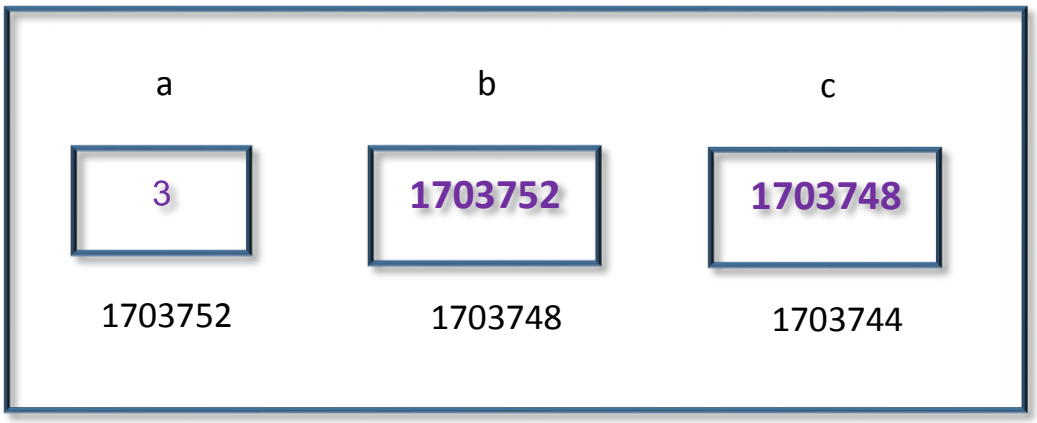
ตัวอย่าง 7.4 โปรแกรม Pointers to Pointers

```c
#include <stdio.h>
int main()
{
    int a = 3;
    int *b;
    int **c;
    b = &a;
    c = &b;
    printf("Address of a = %u  \n" , &a) ;
    printf("Address of a = %u  \n" , b) ;
    printf("Address of b = %u  \n" , &b) ;
    printf("Address of b = %u  \n" , c) ;
    printf("Address of b = %u  \n" , &b) ;
    printf("Value of b = %u  \n" , b) ;
    printf("Value of a = %d  \n" , c) ;
    printf("Value of a = %d  \n" , a) ;
    printf("Value of a = %d  \n" , * (&a) ) ;
    printf("Value of a = %d  \n" , *b) ;
    printf("Value of a = %d  \n" , **c) ;
}
```

หน้าจอแสดงผลลัพธ์



ซึ่งผลที่ได้จะเป็นไปตามรูป



ค่าที่จัดเก็บในตัวแปร a,b และ c และตำแหน่งแอดเดรสที่ใช้จัดเก็บค่าตัวแปรทั้งสาม

และด้วยประสิทธิภาพของตัวแปรพอยน์เตอร์นี้เอง จึงมีการนำพอยน์เตอร์ไปใช้งานกับโครงสร้างข้อมูลไม่ว่าจะเป็นเรื่องของสแตก (Stack) คิว (Queue) และลิงก์ลิงต์ (Linked List)

อาร์เรย์ (Arrays)

การเขียนโปรแกรมเพื่อใช้งาน บ่อยครั้งทีเดียวที่จำเป็นต้องประมวลผลกลุ่มชุดข้อมูล เช่น X1,X2,X3,…..,Xn ตัวอย่างเช่น คะแนนสอบกลางภาค (mid) , คะแนนปลายภาค (fin) , และคะแนนปฏิบัติ (Workshop) ของนักศึกษาห้องหนึ่งที่มีจำนวนทั้งสิ้น 40 คนและหากยังคงใช้การประกาศตัวแปรในรูปแบบเต็มๆ เราจำเป็นต้องประกาศชื่อตัวแปรทั้งหมดเหล่านี้ให้แตกต่างกัน ตามรายละเอียดดังนี้

การประกาศตัวแปรเพื่อจัดเก็บคะแนนกลางภาคของนักศึกษา 40 คน

```
mid1 , mid2 , mid3 , mid4 , mid5 , …….. , mid40 ,
```

การประกาศตัวแปรเพื่อจัดเก็บคะแนนปลายภาคของนักศึกษา 40 คน

```
fin1 , fin2 , fin3 ,fin4,fin5,……fin40
```

การประกาศตัวแปรเพื่อจัดเก็บคะแนนปฏิบัติการของนักศึกษา 40 คน

```
workshop1 , workshop2 , workshop3 , workshop4,…….,workshop40
```

จะพบว่าวิธีการประกาศตัวแปรดังกล่าว จะต้องประกาศชื่อตัวแปรให้มีความแตกต่างกันรวมทั้งสิ้นถึง 120 ตัวด้วยกัน ซึ่งมิใช่เป็นวิธีที่ดีเลย อีกทั้งยังทำให้การอ้างอิงตัวแปรภายในโปรแกรมมีความยุ่งยาก เกิดข้อผิดพลาดได้ง่าย ดังนั้น “ตัวแปรอาร์เรย์” จึงถูกนำมาใช้เพื่อรองรับชุดข้อมูลเหล่านี้

อาร์เรย์เป็นตัวแปรที่ใช้จัดเก็บข้อมูลชนิดเดียวกัน ถือเป็นโครงสร้างข้อมูลชนิดหนึ่งที่มีวิธีการจัดเก็บข้อมูลในรูปแบบแถวลำดับ ในภาษา C เมื่อมีการอ้างอิงตัวแปรอาร์เรย์เมื่อใด เราจะต้องรู้ตำแหน่งของแอดเดรสแรกที่ถูกนำมาจัดเก็บข้อมูลสมาชิกได้ทันที ส่วนการอ้างอิงสมาชิกใดๆในอาร์เรย์ จะใช้ซับสคริปต์ (Subscript) หรือบางครั้งอาจเรียกว่าดรรชนี (Index) เช่น mid[5] หมายความว่าอาร์เรย์ชื่อ mid ที่ตู้หน่วยความจำ 5 ดังนั้น เมื่อนำตัวแปรอาร์เรย์มาใช้ เพื่อจัดเก็บคะแนนสอบตามตัวอย่างข้างต้นที่ผ่านมา เราก็สามารถอ้างอิงได้ดังนี้

mid[1] , mid[2] , [mid3] , [mid4] , …….. , mid[39] ,

fin[1] , fin[2] , [fin3] ,fin[4],fin[5],……fin[39]

Workshop[1] , workshop[2] , workshop[3] ,……,workshop[40]

ครั้นได้นำอาร์เรย์มาใช้กับชุดข้อมูลดังกล่าวย่อมทำให้เราอ้างอิงชื่อตัวแปรดังกล่าวได้สะดวกและง่ายแทนที่จะต้องประกาศตัวแปรให้มีชื่อแตกต่างกันถึง 120 ตัว ก็จะมีเพียง 3 ตัวแปรเท่านั้นคือ mid,fin และ workshop ส่วนการชี้ระบุถึงสมาชิกภายในอาร์เรย์นั้นๆ ก็จะอ้างผิงผ่านเลขดรรชนีหรือขับสคริปต์แทนโดยขับสคริปต์ของอาร์เรย์ในภาษา C จะเริ่มต้นที่ศูนย์

อาร์เรย์หนึ่งมิติ (One Dimension Arrays)

สมาชิกแต่ละตัวภายในอาร์เรย์จะเรียกว่า “อิลิเมนต์ (Element)” ส่วนการอ้างอิงสมาชิกตัวใดตัวหนึ่งจะใช้ขับสคริปต์เป็นตัวชี้ตำแหน่ง นอกจากนี้เรายังสามารถประกาศตัวแปรอาร์เรย์ให้เป็นแบบมิติเดียวหรือหลายมิติก็ได้แต่สำหรับหัวข้อนี้จะขอกล่าวถึงอาร์เรย์หนึ่งมิติก่อน

รูปแบบ : storage_class data_type array [expression] ;

โดยที่ : storage_class คือแบบของตัวแปร

Data_type คือชนิดข้อมู

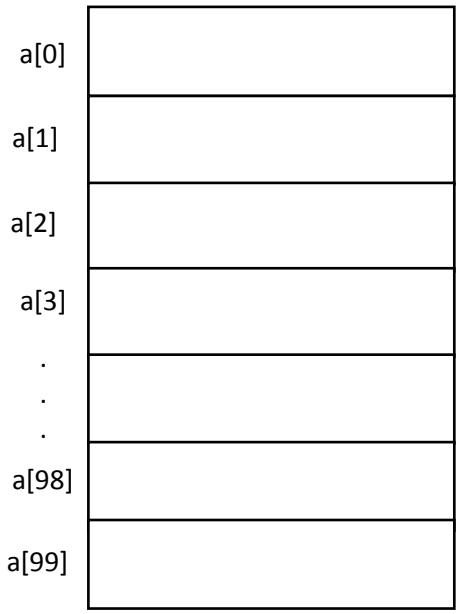
Array คือชื่อตัวแปรอาร์เรย์

Expression คือเลขจำนวนเต็มบวก ที่นำมากำหนดจำนวนสมาชิกของอาร์เรย์ซึ่งอาจจะเป็นค่าคงที่ หรือ นิพจน์ทางคณิตศาสตร์ก็ได้

ตัวอย่างเช่น :

```
char text[80]

static char message [25] ;

static float n [12]



#define SIZE  100

int a[SIZE];

Float num[SIZE * 2];
```

เมื่ออาร์เรย์ในภาษา C เริ่มต้นที่ตำแหน่ง 0 ดังนั้น ถ้าประกาศให้ int a [100]; จึงหมายถึงกำหนดให้ a เป็นอาร์เรย์จัดเก็บสมาชิก 100 ตัว โดยเริ่มต้นที่ 0-99 นั่นเอง



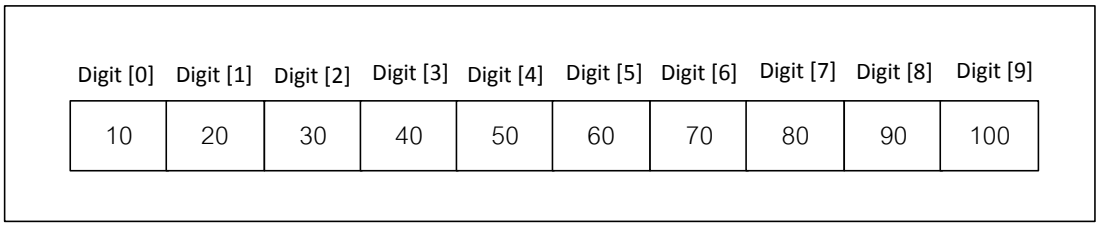
การประกาศตัวแปรอาร์เรย์ a[100] เปรียบเสมือนการจองตู้หน่วยความจำ 100 ตู้

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

เมื่อประกาศตัวแปรอาร์เรย์แล้วหากต้องการกำหนดค่าเริ่มต้นให้กับชุดข้อมูลเหล่านี้ ก็สามารถทำได้ ตัวอย่างเช่น ได้ประกาศตัวแปรอาร์เรย์หนึ่งมิติชื่อ digit โดยให้มีสมาชิกทั้งสิ้น 10 ตัว พร้อมกำหนดชุดข้อมูลเพื่อเป็นค่าเริ่มต้นดังนี้

```
int digit[ 10 ] = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100} ;
```

จากชุดคำสั่งดังกล่าวส่งผลให้สมาชิกในอาร์เรย์ digit มีค่าตามรูป



ค่าเริ่มต้นที่ถูกกำหนดลงในสมาชิกแต่ละตัวของอาร์เรย์ digit

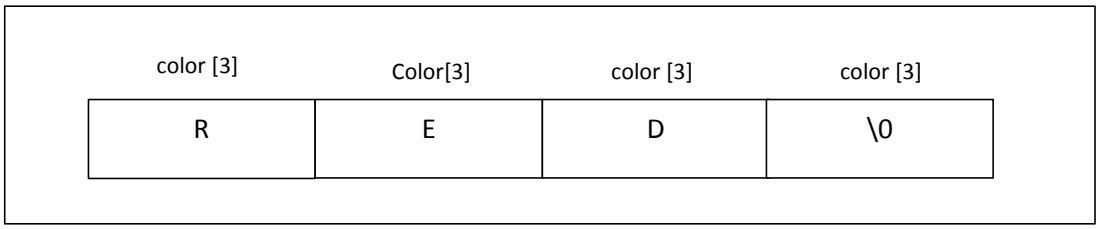
อย่างไรก็ตาม การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ ภาษา C ยังอนุญาตให้เราไม่ต้องระบุขนาดลงไปก็ได้ โดยขนาดของอาร์เรย์จะเท่ากับจำนวนสมาชิกที่ระบุไว้โดยอัตโนมัติ ซึ่งพิจารณาได้จากชุดคำสั่งต่อไปนี้ ผลลัพธ์ที่ได้จะเหมือนกับรูปด้านบน

```
int  digit[  ] = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100} ;
```

กรณีประกาศตัวแปรอาร์เรย์ เพื่อจัดเก็บข้อความ

```
char  color [4]  = {'R' , 'E' , 'D' , '\O'}
```

ค่าที่บรรจุอยู่ในตู้หน่วยความจำของอาร์เรย์ color จะได้ผลลัพธ์ดังรูป



อาร์เรย์ color ที่แต่ละตู้หน่วยความจำ จะจัดเก็บตัวอักขระเพียงหนึ่งตัวเท่านั้น

ในการประกาศตัวแปรชนิดข้อความให้กับอาร์เรย์ชื่อ color นั้นจะมีการบรรจุรหัสพิเศษเพิ่มเติมเข้าไปที่ท้ายอาร์เรย์ นั่นก็คือ "Null Character" ที่แทนด้วยรหัส \0 สำหรับรหัสดังกล่าวจะบอกให้ทราบถึงตำแหน่งสิ้นสุดของข้อความ ทั้งนี้ เมื่อมีการสั่งพิมพ์ข้อความดังกล่าว รหัส \0 จะไม่ถูกแสดงผลออกมาให้เห็นทางจอภาพอย่างไรก็ตาม รายละเอียดเกี่ยวกับข้อความสตริง จะขอกล่าวในหัวข้อของการจัดการข้อมูลสตริง

หน่วยการเรียนรู้ที่ ๘ ฟังก์ชั่นและไลบรารี่

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์
ว ๔.๒ ม.๒/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงคำนวณในการแก้ปัญหา หรือการทำงานที่พบในชีวิตจริง
ว ๔.๒ ม.๒/๒	ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความเข้าใจเกี่ยวกับฟังก์ชั่นและไลบรารี่

๒. เรียนรู้และทำความเข้าใจเกี่ยวกับฟังก์ชั่นที่สร้างขึ้นเอง

๓. เรียนรู้และทำความเข้าใจเกี่ยวกับฟังก์ชั่นมาตรฐานของภาษาซี

<u>สาระการเรียนรู้</u>

- **ความรู้**

๑. ความรู้เกี่ยวกับฟังก์ชั่นและไลบรารี่

๒. ความรู้เกี่ยวกับฟังก์ชั่นที่สร้างขึ้นเอง

๓. ความรู้เกี่ยวกับฟังก์ชั่นมาตรฐานของภาษาซี

- **ทักษะ / กระบวนการ**

๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง

๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา

๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 8 ฟังก์ชันและไลบรารี่

โปรแกรมภาษาซีทุกโปรแกรมจะต้องมีฟังก์ชัน main() เสมอ ในการเขียนโปรแกรมถ้าหากในฟังก์ชัน main() ไม่มีการคืนค่ากลับ โปรแกรมก็ยังคงทำงานได้ แต่ในการเขียนโปรแกรมที่ดีนั้นใน main() ควรมีการคืนค่าด้วย โปรแกรมคอมพิวเตอร์ส่วนใหญ่ที่ใช้งานจริง มักจะเป็นโปรแกรมขนาดใหญ่ เกินกว่าที่จะเขียนรวมๆ กันในฟังก์ชันหลัก main() จึงต้องสร้างฟังก์ชันขึ้นมาเองเพื่อสะดวกในการเรียกใช้งานและลดเวลาในการเขียนโปรแกรม

ความหมายของฟังก์ชัน

ฟังก์ชัน (Function) คือ ชุดคำสั่งที่เขียนขึ้นเพื่อสั่งให้คอมพิวเตอร์ทำงาน ไม่ว่าจะเป็นการรับข้อมูล การประมวลผลหรือการแสดงผลข้อมูล ซึ่งโดยปกติแล้วฟังก์ชันหรือโปรแกรมย่อย จะถูกออกแบบให้ทำงานหน้าที่ใดหน้าที่หนึ่งโดยเฉพาะ ครั้นเมื่อทำงานเสร็จก็จะกลับไปยังฟังก์ชันเพื่อประมวลผลคำสั่งถัดไป สามารถเรียกใช้ฟังก์ชันเพื่อทำงานได้บ่อยครั้งตามต้องการ และฟังก์ชันที่ถูกเรียกใช้ก็ยังสามารถเรียกใช้ฟังก์ชันอื่นๆ ได้อีกตามต้องการ โดยกลุ่มของคำสั่งที่เขียนจะอยู่ภายในเครื่องหมาย { } เพื่อบอกขอบเขต และมีการตั้งชื่อให้กับกลุ่มคำสั่งนั้น เพื่อความสะดวกในการเรียกใช้งาน

ถ้าต้องการให้โปรแกรมทำงานใดๆ ฟังก์ชัน main() จะไปเรียกใช้ฟังก์ชันอื่นๆ ให้โปรแกรมทำงาน ในภาษาซีมีไลบราลีฟังก์ชัน (library function) สำหรับเก็บค่าฟังก์ชันต่างๆ ให้เราเลือกใช้งานอยู่ในไฟล์นามสกุล .h หรือที่เรียกว่า เฮดเดอร์ไฟล์ (header file) เช่น ฟังก์ชัน printf() จะเก็บไว้ในเฮดเดอร์ไฟล์ stdio.h , ฟังก์ชัน getch()จะเก็บไว้ในเฮดเดอร์ไฟล์ conio.h

ประเภทของฟังก์ชัน

ภาษา C เตรียมฟังก์ชันมาตรฐานโดยบรรจุไว้ในคลัง (Library Functions) จำนวนมากพอต่อการใช้งานทั่วไป ซึ่งในการเรียกใช้งานผู้เขียนโปรแกรมต้องรู้ว่าฟังก์ชันดังกล่าวถูกประกาศอยู่ในเฮดเดอร์ใด จากนั้นให้ผนวกเฮดเดอร์ไฟล์ดังกล่าวไว้ที่ส่วนหัวของโปรแกรม และนอกจากฟังก์ชันมาตรฐานแล้ว ในภาษา C ยังอนุญาตให้เราสร้างฟังก์ชันเพื่อใช้งานเอง สำหรับฟังก์ชันที่ใช้งานในภาษา C นั้นมีอยู่ 2 ประเภท

1. ฟังก์ชันมาตรฐาน (Standard Function) เป็นฟังก์ชันที่ภาษา C จัดเตรียมไว้ให้อยู่แล้ว โดยจัดเก็บไว้ในคลัง เมื่อต้องการเรียกใช้ก็ให้ผนวกเฮดเดอร์ไฟล์ไว้ที่ส่วนหัวโปรแกรม

2. ฟังก์ชันที่สร้างขึ้นเอง (User-Defined Functions) เป็นฟังก์ชันที่ผู้เขียนโปรแกรมสร้างขึ้นเองเพื่อทำงานใดงานหนึ่งโดยเฉพาะ หรืออาจเป็นฟังก์ชันที่สร้างขึ้นเพื่อเป็นส่วนหนึ่งของโปรแกรมหลัก ที่เรียกว่าโปรแกรมย่อย โดยมีวัตถุประสงค์เพื่อต้องการแบ่งส่วนโปรแกรมออกมาทำงานต่างหาก ลดความซ้ำซ้อน

รูปแบบของฟังก์ชัน

รูปแบบของฟังก์ชันของภาษา ซี มีอยู่ 4 รูปแบบ

แบบที่ 1 ฟังก์ชันแบบไม่มีการส่งค่ากลับ และไม่มีพารามิเตอร์ เป็นฟังก์ชันที่ไม่มีการส่งค่ากลับไปให้กับฟังก์ชันที่เรียกมา และไม่มีการส่งค่าจากฟังก์ชันที่เรียกมาให้ด้วย

แบบที่ 2 ฟังก์ชันแบบไม่มีการส่งค่ากลับ และพารามิเตอร์ เป็นฟังก์ชันที่จะไม่มีการส่งค่ากลับไปให้ฟังก์ชันที่เรียกขึ้นมา แต่มีการส่งค่าจากฟังก์ชันที่เรียกมาให้ด้วย

แบบที่ 3 ฟังก์ชันแบบมีการส่งค่ากลับ และไม่มีพารามิเตอร์ เป็นฟังก์ชันที่จะมีการส่งค่ากลับไปให้ฟังก์ชันที่เรียกมา แต่ไม่มีการส่งค่าจากฟังก์ชันที่เรียกมาให้ด้วย

แบบที่ 4 ฟังก์ชันแบบมีการส่งค่ากลับ และมีพารามิเตอร์ เป็นฟังก์ชันที่จะมีการส่งค่ากลับไปให้กับฟังก์ชันที่เรียกมา แต่มีการส่งค่าจากฟังก์ชันที่เรียกมาให้ด้วย

ฟังก์ชันที่สร้างขึ้นเอง (User-Defined Functions)

เป็นฟังก์ชันที่ผู้เขียนโปรแกรมสร้างขึ้นเองเพื่อทำงานใดงานหนึ่งโดยเฉพาะ หรืออาจเป็นฟังก์ชันที่สร้างขึ้นเพื่อเป็นส่วนหนึ่งของโปรแกรมหลัก ที่เรียกว่าโปรแกรมย่อย โดยมีวัตถุประสงค์เพื่อต้องการแบ่งส่วนโปรแกรมออกมาทำงานต่างหาก เพื่อลดความซ้ำซ้อนและง่ายต่อการตรวจสอบแก้ไข

ในการสร้างฟังก์ชันเพื่อใช้งานเอง จะมีส่วนประกอบสำคัญอยู่ 2 ส่วนด้วยกันคือ

1. ส่วนหัวของฟังก์ชัน เป็นบรรทัดแรกที่ใช้สำหรับนิยามฟังก์ชัน ประกอบด้วยชนิดข้อมูลที่คืนค่ากลับมา ถัดไปก็คือชื่อฟังก์ชันและถัดไปอีกเป็นกลุ่มของอาร์กิวเมนต์ที่อยู่ในเครื่องหมายวงเล็บ โดยสามารถมีหลายอาร์กิวเมนต์ด้วยการใช้เครื่องหมาย , เพื่อแบ่งอาร์กิวเมนต์แต่ละตัว อย่างไรก็ตามอาร์กิวเมนต์เป็นเพียงทางเลือก อาจมีหรือไม่มีก็ได้

รูปแบบ :

```
data-type function(type_1 arg_1 , type_2 arg_2 , ….. , type_n arg_n);
```

รูปแบบ :

```
ชนิดข้อมูล  ชื่อฟังก์ชัน(อาร์กิวเมนต์ 1 , อาร์กิวเมนต์ 2, ….. , อาร์กิวเมนต์ n);
```

ตัวอย่าง :

```
int count(int x , int y , int z);
```

อธิบายได้ว่า :

- ชนิดข้อมูลที่คืนค่ากลับมา ในที่นี้คือเลขจำนวนเต็ม (integer)
- ฟังก์ชันนี้มีชื่อว่า count
- ฟังก์ชันนี้มีการรับค่าอาร์กิวเมนต์อยู่ 3 ตัวแปร ซึ่งทั้งหมดมีชนิดข้อมูลเป็นเลขจำนวนเต็ม

ในการเขียนฟังก์ชันขึ้นมาใช้งานอย่างใดอย่างหนึ่ง เราสามารถจำแนกฟังก์ชันที่เขียนขึ้นตามลักษณะการส่งค่าไปและรับค่ากลับได้ 3 แบบ คือ

1. ฟังก์ชันที่ไม่มีการส่งค่าไปและรับค่ากลับ เช่น ให้หาค่าที่เข้ามาทางอินพุต

2. ฟังก์ชันที่มีการส่งค่าไปแต่ไม่มีรับค่ากลับ เช่น สั่งให้พิมพ์ข้อความออกทางหน้าจอ

3. ฟังก์ชันที่มีทั้งการส่งค่าไปและรับค่ากลับ เช่น ส่งค่าไปให้คำนวณ แล้วรับค่าที่คำนวณแล้วกลับ

ซึ่งฟังก์ชันแต่ละแบบก็เหมาะกับงานแต่ละอย่าง ดังนั้นผู้เขียนฟังก์ชันจึงจำเป็นที่จะต้องศึกษาทำความเข้าใจฟังก์ชันแต่ละแบบ เพื่อจะได้มาประยุกต์ใช้กับงานได้อย่างเหมาะสม

2. ตัวฟังก์ชัน คือกลุ่มของชุดคำสั่งที่อยู่ภายในฟังก์ชัน และที่ท้ายโปรแกรมจะต้องมีการคืนค่ากลับไปยังผู้เรียก จะสามารถคืนค่าได้เพียงค่าเดียวเท่านั้น

รูปแบบ :

```
return expression;
```

ตัวอย่าง :

```
return count;      //คืนค่า count กลับไป
return (count + 2);      //คืนค่าตัวแปร count บวก 2 กลับไป
return;      //กลับไปยังฟังก์ชันที่เรียก โดยไม่มีการคืนค่าใดๆ
```

ตัวอย่าง การส่งผ่านค่าไปยังฟังก์ชันที่สร้างขึ้นเอง

เมื่อมีการเรียกใช้ฟังก์ชัน กลไกการรับส่งข้อมูลไปยังฟังก์ชันคืออาร์กิวเมนต์ (อาจเรียกว่าพารามิเตอร์)

```
1  #include<stdio.h>
2  int count(int x, int y, int z )
3  {
4      int sum;
5      sum = x + y + z;
6      return(sum);
7  }
8  main()
9  {
10     int x,y,z,sum;
11     printf("Enter number1 : ");
12     scanf("%d",&x);
13     printf("Enter number2 : ");
14     scanf("%d",&y);
15     printf("Enter number3 : ");
16     scanf("%d",&z);
17     sum = count(x,y,z);
18     printf("Sum = %d\n" , sum);
19 }
```

การประกาศค่าอาร์กิวเมนต์ เขียนได้ 2 รูปแบบ

1. int count(int x, int y, int z)

2. int count(x, y, z)
   int x,y,z;

คำอธิบายโปรแกรม

แม้ว่าฟังก์ชันที่สร้างขึ้น จะวางตำแหน่งไว้ก่อนฟังก์ชัน main() แต่ก็ไม่ได้หมายความว่าจะถูกประมวลผลก่อน เนื่องจากกลไกการสั่งงานของโปรแกรมทั้งหมด จะอยู่ภายใต้การควบคุมของฟังก์ชัน main()

<u>ส่วนของฟังก์ชัน count</u>

บรรทัดที่ 2 ฟังก์ชันที่สร้างเองชื่อ count โดยมีอาร์กิวเมนต์ 3 ค่าด้วยกันคือ x,y,z มีชนิดข้อมูลเป็นเลขจำนวนเต็ม int ทั้งหมด และคืนค่ากลับไปเป็นเลขจำนวนเต็ม

บรรทัดที่ 3 คำนวณหาผลรวม แล้วเก็บไว้ที่ตัวแปร sum

บรรทัดที่ 4 ส่งค่า sum กลับไป

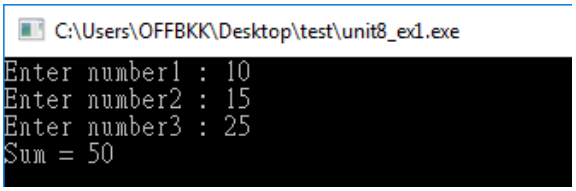
<u>ส่วนของฟังก์ชัน main()</u>

บรรทัดที่ 12-16 รอรับค่า x,y,z

บรรทัดที่ 17 เรียกใช้ฟังก์ชัน count ด้วยการส่งผ่านค่า x,y,z ลงไป แล้วให้ผลลัพธ์คืนค่ากลับมายังตัวแปร sum

บรรทัดที่ 18 พิมพ์ผลรวม

ผลลัพธ์ของโปรแกรม



การเข้าถึงฟังก์ชัน (Accessing a Function)

ภาษา C มีรูปแบบการเข้าถึงฟังก์ชัน 4 รูปแบบ

1. ฟังก์ชันที่ไม่มีการส่งผ่านค่าใดๆ

เป็นฟังก์ชันที่ไม่มีการส่งผ่านค่าใดๆ ชุดคำสั่งจะมุ่งเน้นการทำงานใดงานหนึ่งเท่านั้น ฟังก์ชันแบบนี้จะไม่มีการส่งค่าไปหรือกลับ โดยการใช้งานจะประกาศ void นำหน้าแล้วตามด้วยชื่อฟังก์ชัน

รูปแบบ :

```
void function_name (void)
{
        statement_1;
        statement_2;
        statement_n;
}
```

ตัวอย่าง ฟังก์ชันที่ไม่มีการส่งผ่านค่าใดๆ

```c
#include <stdio.h>

void star(void);

main()
{
    star();
    printf("\t\t C Programming \n");
    star();
}

void star()
{
    for(int i = 0; i<= 50; i++)
        printf("*");
    printf("\n");
}
```

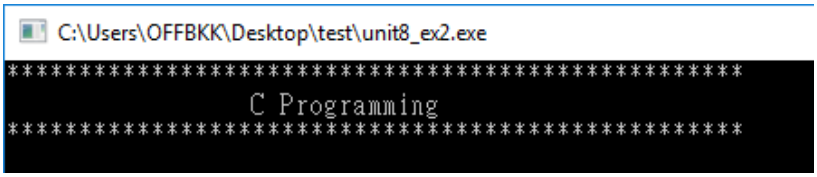
คำอธิบายโปรแกรม

บรรทัดที่ 3 ประกาศฟังก์ชันต้นแบบชื่อ star

บรรทัดที่ 7-9 เรียกใช้ฟังก์ชัน star()

บรรทัดที่ 12-17 ฟังก์ชันที่สร้างขึ้นเองชื่อ star() ไม่มีการส่งผ่านค่าใดๆเข้ามาหรือกลับไป ทำหน้าที่แค่แสดงผล * เท่านั้น

ผลลัพธ์ของโปรแกรม



2. ฟังก์ชันที่ส่งผ่านค่าทางเดียว

เป็นฟังก์ชันที่มีการรับค่าอาร์กิวเมนต์เข้ามายังฟังก์ชัน และภายหลังเสร็จสิ้นการทำงานของตัวฟังก์ชันจะไม่มีการคืนค่าใดๆ กลับไป

รูปแบบ :

```
void function_name (type_1 arg_1, type_2 arg_2, .... type_n arg_n)
{
        statement_1;
        statement_2;
        statement_n;
}
```

ตัวอย่าง ฟังก์ชันที่ส่งผ่านค่าทางเดียว

```
 1  #include <stdio.h>
 2
 3  void bar(int);
 4
 5  main()
 6  {
 7      int female,male;
 8      printf("Enter amount of Female => ");
 9      scanf("%d",&female);
10      printf("Enter amount of male => ");
11      scanf("%d",&male);
12      printf("\n");
13      bar(female);
14      bar(male);
15  }
16
17  void bar(int amount)
18  {
19      for(int i=1; i <= amount; i++)
20          printf("*");
21      printf("%d \n",amount);
22  }
```

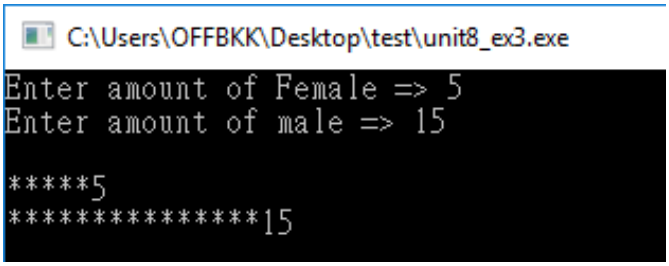
คำอธิบายโปรแกรม

บรรทัดที่ 3 ประกาศฟังก์ชันต้นแบบชื่อ bar

บรรทัดที่ 13-14 เรียกใช้ฟังก์ชัน bar() โดยส่งผ่านค่า female และ male ลงไป

บรรทัดที่ 17-22 เป็นส่วนของฟังก์ชัน bar ซึ่งเป็นฟังก์ชันที่สร้างขึ้นเอง ทำหน้าพิมพ์ * พร้อมทั้งค่าที่ถูกส่งกลับมา นั่นคือ female และ male

ผลลัพธ์ของโปรแกรม



3. ฟังก์ชันที่ส่งผ่านค่าทั้งไปและกลับ

เป็นฟังก์ชันที่นอกจากรับค่าอาร์กิวเมนต์เข้ามายังฟังก์ชันแล้ว ยังมีการคืนค่าผลลัพธ์กับไปยังฟังก์ชัน

รูปแบบ :

```
data_type function_name (type_1 arg_1, type_2 arg_2, .... type_n arg_n)
{
        statement_1;
        statement_2;
        statement_n;
        return(expression);
}
```

ตัวอย่าง ฟังก์ชันที่ส่งผ่านค่าทั้งไปและกลับ (คำนวณหาค่าแฟคทอเรียล)

```
1   #include <stdio.h>
2
3   int Factorial(int);
4
5   int main()
6   {
7       int Number = 5;
8       printf("%d face is %d", Number, Factorial(Number));
9       return 0;
10  }
11
12  int Factorial(int n)
13  {
14      int result = 1, i;
15      for (i = 1; i <= n; i++)
16      {
17          result *= i;
18      }
19      return result;
20  }
```

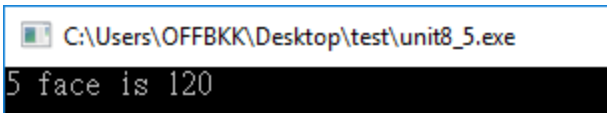
คำอธิบายโปรแกรม

บรรทัดที่ 3 ประกาศฟังก์ชันต้นแบบชื่อ Factorial

บรรทัดที่ 13-14 เรียกใช้ฟังก์ชัน Factorial โดยส่งผ่านค่า Number ลงไป

บรรทัดที่ 17-22 เป็นส่วนของฟังก์ชัน Factorial ซึ่งเป็นฟังก์ชันที่สร้างขึ้นเอง หลังจากประมวลชุดคำสั่งในฟังก์ชันแล้ว Factorial() จะคืนค่า result กลับไป

ผลลัพธ์ของโปรแกรม



4. ฟังก์ชันที่คืนค่ากลับไปเท่านั้น

เป็นฟังก์ชันที่ไม่มีการรับค่าอาร์กิวเมนต์ใดๆ เข้ามายังฟังก์ชัน แต่ภายหลังจากการประมวลผลชุดคำสั่งภายในฟังก์ชันเสร็จสิ้นแล้ว จะมีการคืนค่ากลับไปยังฟังก์ชันที่เรียกใช้

รูปแบบ :

```
data_type function (void)
{
        statement_1;
        statement_2;
        statement_n;
        return (expression);
}
```

ตัวอย่าง ฟังก์ชันที่คืนค่ากลับไปเท่านั้น

```
1   #include <stdio.h>
2   #include <conio.h>
3
4   int getChar(void);
5
6   main()
7   {
8       char ch;
9       printf("Enter any character => \n");
10      ch = getChar();
11      printf("%c is %d for ASCII valua\n",ch,ch);
12  }
13  int getChar (void)
14  {
15      char a;
16      a = getch();
17      return a;
18  }
```

คำอธิบายโปรแกรม

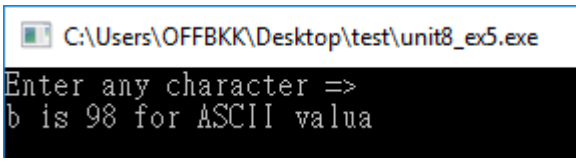
บรรทัดที่ 4 ประกาศฟังก์ชันต้นแบบชื่อ getChar ที่มีการคืนค่า int กลับไป

บรรทัดที่ 10 เรียกใช้ฟังก์ชัน getChar โดยไม่มีการส่งผ่านค่าใดๆลงไป แต่ผลลัพธ์ที่คืนกลับมา จะถูกเก็บไว้ที่ตัวแปร a

บรรทัดที่ 16 รับค่าจากแป้นพิมพ์ใดๆ ด้วยฟังก์ชัน getch() และเก็บไว้ในตัวแปร a

บรรทัดที่ 17 คืนค่า a กลับไปยังฟังก์ชันที่เรียกใช้ โดยค่า a ที่คืนค่ากลับไป จะถูกส่งผ่านคืนกลับไปยังตัวแปร ch ที่อยู่ในฟังก์ชัน main()

ผลลัพธ์ของโปรแกรม



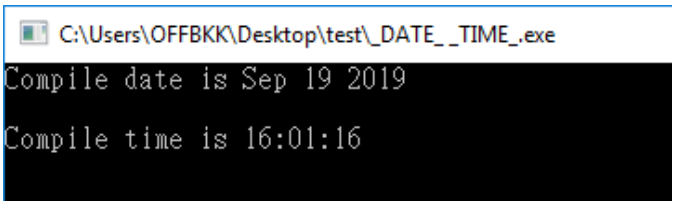
ฟังก์ชันมาตรฐานที่ประกาศไว้ในเฮดเดอร์ไฟล์ time.h

สำหรับฟังก์ชันที่ประกาศไว้ในเฮดเดอร์ไฟล์ time.h เป็นฟังก์ชันแสดงผลวันที่และเวลาของระบบ ภาษาC ไม่ได้เตรียมฟังก์ชันเหล่านี้มาให้ใช้งานโดยตรง ดังนั้นหากต้องการใช้ ต้องสร้างเอง

ตัวอย่าง การเรียกใช้มาโคร DATE และ TIME เพื่อแสดงวันที่คอมไพล์โปรแกรม

```
1  #include <stdio.h>
2
3  main()
4  {
5      printf("Compile date is %s\n\n",__DATE__);
6      printf("Compile time is %s\n\n",__TIME__);
7  }
```

ผลลัพธ์ของโปรแกรม



ฟังก์ชันมาตรฐาน (Standard function)

อาจจะเรียกฟังก์ชันมาตรฐานว่า “ไลบรารีฟังก์ชัน” (library functions) ฟังก์ชันมาตรฐานจะถูกเก็บไว้ในไลบรารี เช่น ฟังก์ชันทางคณิตศาสตร์ ฟังก์ชันเกี่ยวกับสตริง ฟังก์ชันเกี่ยวกับการเปรียบเทียบ ฟังก์ชันเกี่ยวกับการแสดงผล และฟังก์ชันเกี่ยวกับวันเวลา เป็นต้น ซึ่งการเรียกฟังก์ชันขึ้นมาใช้นั้น ต้องทราบก่อนว่าฟังก์ชันนั้นเก็บอยู่ในไลบรารีใด จากนั้นเรียกไลบรารีผ่านคำสั่ง #include แล้วตามด้วยชื่อไลบรารี่นั้น ที่ส่วนหัวของโปรแกรม เช่น #include<stdio.h>

ผู้เขียนโปรแกรมต้องเข้าใจว่าฟังก์ชันที่จะใช้นั้นต้องรับค่าใด แล้วจะคืนค่าใดออกมา ตัวอย่าง การใช้ฟังก์ชันทางคณิตศาสตร์ โดยการเรียกใช้ฟังก์ชันเหล่านี้จะต้องเรียก library math.h ขึ้นมาก่อน ยกตัวอย่างเช่น

ฟังก์ชัน pow(x,y) คือ ฟังก์ชันที่ใช้หาค่ายกกำลังของ xy โดยที่ตัวแปร x และตัวแปร y มีชนิดเป็น double ซึ่งฟังก์ชัน pow(x,y) จะถูกเก็บไว้ใน header file ที่ชื่อว่า math.h ดังนั้นจึงต้องใช้คำสั่ง #include<math.h> แทรกอยู่ในส่วนตอนต้นของโปรแกรมเหนือฟังก์ชัน main( ) จึงจะสามารถเรียกใช้ฟังก์ชัน pow(x,y) มาใช้งานภายในโปรแกรมนี้ได้

ฟังก์ชันมาตรฐาน เฉพาะที่จำเป็นและเรียกใช้งานบ่อยๆ มีดังรายละเอียด header file ต่อไปนี้

1. ฟังก์ชันเกี่ยวกับการแสดงผลทางจอภาพ

- **ไลบรารี conio.h เกี่ยวกับการแสดงผลทางจอภาพ** มีฟังก์ชันที่ใช้ดังนี้

  - ฟังก์ชัน getchar() ใช้ในการรับข้อมูล 1 อักขระ โดยการกด Enter

  - ฟังก์ชัน getche() ใช้ในการรับข้อมูล 1 อักขระ โดยไม่ต้องกด Enter

  - ฟังก์ชัน getch() ใช้ในการรับข้อมูล 1 อักขระไม่ปรากฏให้เห็นในการรับข้อมูล

  - ฟังก์ชัน putchar() ใช้ในการรับข้อมูล 1 อักขระออกทางจอภาพ

  - ฟังก์ชัน clrscr() ใช้ในการลบจอภาพ

  จะใช้คำสั่ง #include <conio.h > แทรกอยู่ตอนต้นของโปรแกรม

- **ไลบรารี stdio.h เกี่ยวกับการแสดงผลทางจอภาพ** มีฟังก์ชันที่ใช้ดังนี้

  - ฟังก์ชัน printf() ใช้ในการแสดงผลข้อมูล

  - ฟังก์ชัน Scanf() ใช้ในการรับข้อมูล

จะใช้คำสั่ง #include < stdio.h> แทรกอยู่ตอนต้นของโปรแกรม

2. ฟังก์ชันทางคณิตศาสตร์ (mathematic functions)

เป็นฟังก์ชันที่ใช้สำหรับการคำนวณทางคณิตศาสตร์ และก่อนที่จะใช้ฟังก์ชันประเภทนี้ จะต้องใช้คำสั่ง #include <math.h> แทรกอยู่ตอนต้นของโปรแกรม และตัวแปรที่จะใช้ฟังก์ชันประเภทนี้จะต้องมีชนิด (type) เป็น double เนื่องจากผลลัพธ์ที่ได้จากฟังก์ชันประเภทนี้จะได้ค่าส่งกลับของข้อมูลเป็น double เช่นกัน

➢ **ไลบรารี math.h เกี่ยวกับทางคณิตศาสตร์** มีฟังก์ชันที่ใช้ดังนี้

- ฟังก์ชัน sqrt() ใช้ในการหาราก (root) ที่สองของเลขจำนวนเต็ม
- ฟังก์ชัน exp(x) เป็นฟังก์ชันที่ใช้หาค่า ex (Exponential)
- ฟังก์ชัน pow(x,y) เป็นฟังก์ชันที่ใช้หาค่ายกกำลังของ xy
- ฟังก์ชัน sin(x) เป็นฟังก์ชันที่ใช้หาค่า sine ของ x
- ฟังก์ชัน cos(x) เป็นฟังก์ชันที่ใช้หาค่า cosine ของ x
- ฟังก์ชัน tan(x) เป็นฟังก์ชันที่ใช้หาค่า tan ของ x
- ฟังก์ชัน log(n) เป็นฟังก์ชันที่ใช้หาค่า log ฐาน n
- ฟังก์ชัน log10(x) เป็นฟังก์ชันที่ใช้หาค่า log ฐาน 10
- ฟังก์ชัน ceil(x) เป็นฟังก์ชันที่ใช้หาค่าปัดเศษทศนิยมของตัวแปร x
- ฟังก์ชัน floor(x) เป็นฟังก์ชันที่ใช้หาค่าตัดเศษทศนิยมทิ้งของตัวแปร x
- ฟังก์ชัน fabs(x) เป็นฟังก์ชันที่ใช้หาค่าสมบูรณ์ (absolute value) x

ตัวอย่าง โปรแกรมแสดงการทำงานของฟังก์ชัน sqrt()การหาราก (root) ที่สองของเลขจำนวนเต็ม

```
1  #include<stdio.h>
2  #include<math.h>
3  main( ){
4      int x,y;
5      printf("Enter number :");
6      scanf("%d",&x);
7      y = sqrt(x);
8      printf("square root of x = %d",y);
9  }
```

คำสั่ง #include <math.h>

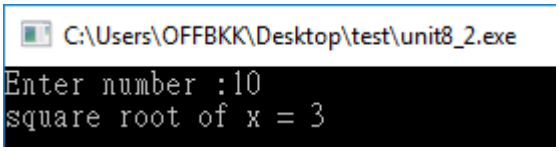
ประกาศฟังก์ชัน sqrt

อธิบายโปรแกรม

ประกาศคำสั่ง #include <math.h> แทรกอยู่ตอนต้นของโปรแกรม

ฟังก์ชัน sqrt() ใช้ในการหาราก (root) ที่สองของเลขจำนวนเต็ม

ผลลัพธ์ของโปรแกรม



3. ฟังก์ชันเกี่ยวกับตัวอักษร (character functions)

เป็นฟังก์ชันที่ใช้กับข้อมูลที่มีชนิดเป็น single char เท่านั้น ใช้คำสั่ง #include<ctype.h> แทรกอยู่ตอนต้นของโปรแกรม จึงจะสามารถเรียกใช้ฟังก์ชันประเภทนี้ได้

- **ไลบรารี ctype.h เกี่ยวกับตัวอักษร** มีฟังก์ชันที่ใช้ดังนี้

- ฟังก์ชัน isalnum(ch) เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลที่อยู่ในตัวแปรมีค่าเป็นตัวอักษรหรือตัวเลข

- ฟังก์ชัน isalpha(ch) เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลที่อยู่ในตัวแปรมีค่าเป็นตัวอักษรหรือไม่

- ฟังก์ชัน isdigit(ch) เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลที่อยู่ในตัวแปรเป็นตัวเลข 0 ถึง 9 หรือไม่

- ฟังก์ชัน islower(ch) เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลที่อยู่ในตัวแปรเป็นตัวเล็กหรือไม่

- ฟังก์ชัน isupper(ch) เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลที่อยู่ในตัวแปรเป็นตัวใหญ่หรือไม่

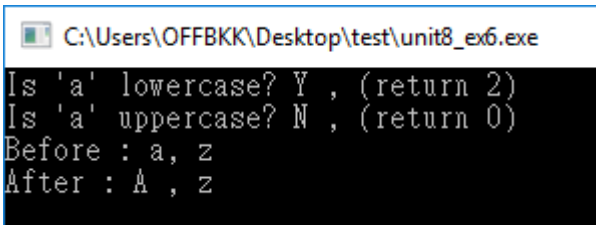
- ฟังก์ชัน tolowre(ch) เป็นฟังก์ชันที่ใช้ในการเปลี่ยนตัวอักษรตัวใหญ่ให้เป็นตัวเล็ก

- ฟังก์ชัน toupper(ch) เป็นฟังก์ชันที่ใช้ในการเปลี่ยนตัวอักษรตัวเล็กให้เป็นตัวใหญ่

ตัวอย่าง การใช้งานฟังก์ชัน islower() , isupper() , tolower() , toupper()

```c
#include <stdio.h>
#include <ctype.h>

 main()
{
    char ch1 = 'a';
    char ch2 = 'z';

    printf("Is '%c' lowercase? Y , (return %d)\n",ch1,islower(ch1));
    printf("Is '%c' uppercase? N , (return %d)\n",ch1,isupper(ch1));
    printf("Before : %c, %c\n", ch1,ch2);
    ch1 = toupper(ch1);
    ch2 = tolower(ch2);
    printf("After : %c , %c\n\n", ch1, ch2);
}
```

ผลลัพธ์ของโปรแกรม



4. ฟังก์ชันเกี่ยวกับสตริง (string functions)

เป็นฟังก์ชันที่ใช้กับข้อมูลชนิดสตริง (string) ซึ่งก็คือข้อความ ฟังก์ชันประเภทนี้จะใช้คำสั่ง #include<string.h> แทรกอยู่ตอนต้นของโปรแกรม

- **ไลบรารี string.h เกี่ยวกับข้อความ** มีฟังก์ชันที่ใช้ดังนี้

- ฟังก์ชัน strlen() ใช้ในการนับความยาวของอักขระที่รับเข้ามา

- ฟังก์ชัน strcpy() ใช้ในการทำสำเนาข้อความจากข้อความหนึ่งไปยังอีกข้อความหนึ่ง

- ฟังก์ชัน strcmp () ใช้ในการเปรียบเทียบข้อความ 2 ข้อความ

- ฟังก์ชัน strcal() ใช้ในการเชื่อมตั้งแต่ 2 ข้อความเข้าด้วยกัน

จะใช้คำสั่ง #include <string.h> แทรกอยู่ตอนต้นของโปรแกรม

5. ฟังก์ชันเกี่ยวกับการแปลงค่าสตริง (string functions)

➢ **ไลบรารี stdlib.h เกี่ยวกับการแปลงค่า string** มีฟังก์ชันที่ใช้ดังนี้

- ฟังก์ชัน atoi(s) เป็นฟังก์ชันที่ใช้ในการแปลงค่า ข้อความ (string) เป็นตัวเลขจำนวนเต็ม (integer)

- ฟังก์ชัน atof(s) เป็นฟังก์ชันที่ใช้ในการแปลงค่า ข้อความ (string) เป็นตัวเลขจำนวนทศนิยม(float)

- ฟังก์ชัน atol(s) เป็นฟังก์ชันที่ใช้ในการแปลงค่า ข้อความ (string) เป็นตัวเลขจำนวนเต็ม (integer) ชนิด long integer

- ฟังก์ชัน rand() เป็นฟังก์ชันที่ใช้สุ่มตัวเลข ในช่วง 0-32,767

- ฟังก์ชัน srand() เป็นฟังก์ชันที่นำมาใช้ร่วมกับฟังก์ชัน rand() เพื่อให้การสุ่มตัวเลขในแต่ละครั้งไม่ให้ซ้ำของเดิม

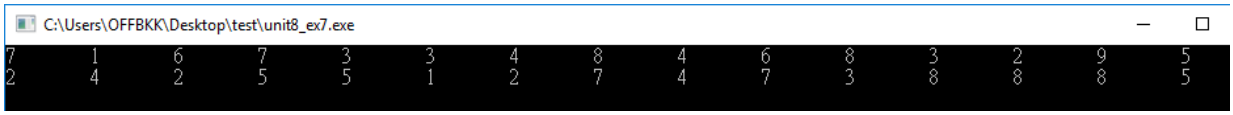
- ฟังก์ชัน system() เป็นฟังก์ชันที่ทำให้เราสามารถสั่งรันโปรแกรมผ่านคำสั่งใน command Line ได้

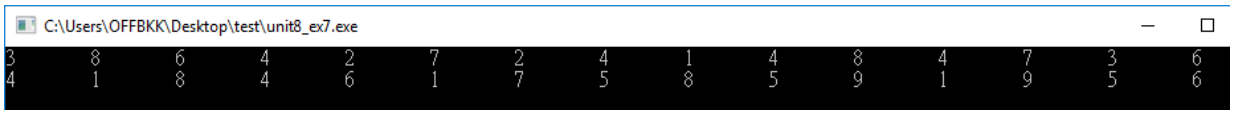
จะใช้คำสั่ง #include< stdlib.h> แทรกอยู่ตอนต้นของโปรแกรม

ตัวอย่าง โปรแกรมสุ่มตัวเลขตั้งแต่ 1 ถึง 9 จำนวน 30 รอบ ด้วยฟังก์ชัน rand() และ srand()

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
main()
{
    int limit = 9;
    srand(time(NULL));
    for (int i = 1; i <= 30; i++)
    printf("%d\t", 1 + (rand() % limit));
}
```

ผลลัพธ์ของโปรแกรม





เมื่อมีการสั่งรันโปรแกรมในแต่ละครั้ง เลขสุ่มที่ได้จะแตกต่างกัน เพราะคำสั่ง srand(time(NULL)); เป็นการอ่านค่านาฬิกาของระบบแล้วส่งผ่านไปยัง seed number โดยอัตโนมัติ จึงส่งผลให้การรันแต่ละครั้ง ได้ค่าเลขสุ่มแตกต่างกัน และเนื่องจากมีการใช้เวลานาฬิกาของระบบ จึงต้องผนวกไฟล์ <time.h>

6. ฟังก์ชันเกี่ยวกับการติดต่อระบบปฏิบัติการ

- **ไลบรารี dos.h เกี่ยวกับการติดต่อระบบปฏิบัติการ** มีฟังก์ชันที่ใช้ดังนี้

  - ฟังก์ชัน gettime() เป็นฟังก์ชันที่ใช้ในการติดต่อเวลาของระบบปฏิบัติการ

  - ฟังก์ชัน getdate() เป็นฟังก์ชันที่ใช้ในการติดต่อวันที่ของระบบปฏิบัติการ

หน่วยการเรียนรู้ที่ ๕ สตริงและสตรักเจอร์

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๒ ม.๑/๒	ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์
ว ๔.๒ ม.๒/๑	ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงคำนวณในการแก้ปัญหา หรือการทำงานที่พบในชีวิตจริง
ว ๔.๒ ม.๒/๒	ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความเข้าใจเกี่ยวกับโครงสร้างข้อมูลชนิดสตรัคเจอร์

๒. เรียนรู้และทำความเข้าใจเกี่ยวกับการประกาศข้อมูลแบบสตรัคเจอร์

<u>สาระการเรียนรู้</u>

- **ความรู้**
  - ๑. ความรู้เกี่ยวกับสตรัคเจอร์
  - ๒. ความรู้เกี่ยวกับสตริง
  - ๓. การประยุกต์เขียนโปรแกรมภาษาซี

- **ทักษะ / กระบวนการ**
  - ๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง
  - ๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา
  - ๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

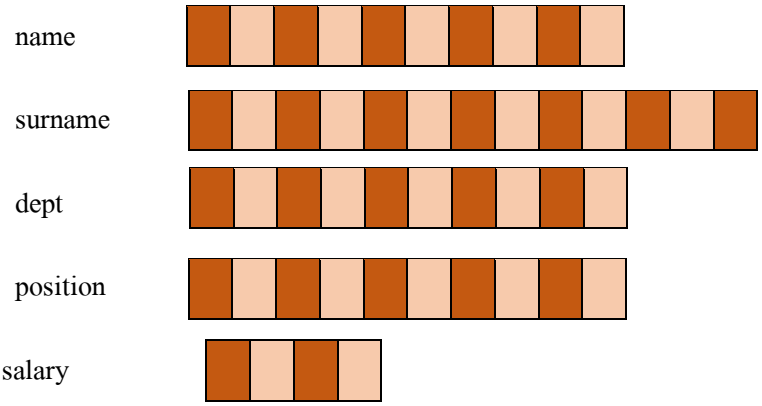
๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 9 สตริงและสตรักเจอร์

สตรัคเจอร์ คือ โครงสร้างข้อมูลที่นำข้อมูลที่มีชนิดข้อมูลแตกต่างกัน แต่มีความสัมพันธ์ของข้อมูลต่อกัน มาเก็บเข้าไว้ภายในโครงสร้างเดียวกัน ตัวอย่างของข้อมูลแบบสตรัคเจอร์ เช่น การเก็บข้อมูลของพนักงานบริษัทแห่งหนึ่งที่ต้องเก็บชื่อ, นามสกุล, แผนก ตำแหน่ง และเงินเดือน โดยการเก็บชื่อ, นามสกุล, แผนก และตำแหน่ง เป็นสตริง และทำการเก็บเงินเดือนเป็นเลขจำนวนจริง

```
Struct employee {
Char name[10];
Char surname[13];
Char dept[10];
Char position[10];
Float salary;
};
struct employee emp;
```



แสดงการเก็บข้อมูลของพนักงานบริษัทในลักษณะของสตรัคเจอร์

ข้อมูลของพนักงานบริษัทแต่ละคน ไม่ว่าจะเป็นชื่อ, นามสกุล, แผนก ตำแหน่ง และเงินเดือน แม้จะมีชนิดข้อมูลที่แตกต่างกันก็ตาม แต่ก็สามารถเก็บไว้ภายในสตรัคเจอร์เดียวกันได้ คือ สตรัคเจอร์ ชื่อ employee

โดย employee คือ ชื่อสตรัคเจอร์

name, surname , dept, position , salary คือสมาชิก (members) ของสตรัคเจอร์

emp คือ ชื่อตัวแปรที่ถูกกำหนดเป็นชนิดข้อมูลแบบสตรัคเจอร์ employee

รูปแบบการประกาศข้อมูลแบบสตรัคเจอร์

การประกาศข้อมูลแบบสตรัคเจอร์สามารถทำได้ 2 รูปแบบ คือ

รูปแบบที่ 1

```
Struct ชื่อของสตรัคเจอร์ {
        ชนิดของข้อมูลตัวที่ 1  ตัวแปรตัวที่ 1;
        ชนิดของข้อมูลตัวที่ 2  ตัวแปรตัวที่ 2;
        ....
        ชนิดของข้อมูลตัวที่ n  ตัวแปรตัวที่ n;
} ชื่อตัวแปรที่ใช้อ้างอิงสตรัคเจอร์;
```

เช่น

```
struct student{
    char student_id[10];
    int points;
    float grade;
}std;
```

รูปแบบที่ 2

Struct ชื่อของสตรัคเจอร์ {

ชนิดของข้อมูลตัวที่ 1 ตัวแปรตัวที่ 1;

ชนิดของข้อมูลตัวที่ 2 ตัวแปรตัวที่ 2;

....

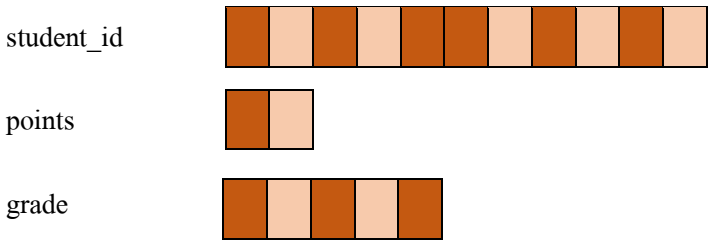
ชนิดของข้อมูลตัวที่ n ตัวแปรตัวที่ n;

} ชื่อของสตรัคเจอร์ ชื่อตัวแปรที่ใช้อ้างอิงสตรัคเจอร์;

เช่น

```
struct student{
    char student_id[10];
    int points;
    float grade;
};
struct student std;
```

การประกาศข้อมูลแบบสตรัคเจอร์ทั้ง 2 รูปแบบ ให้ความหมายที่เหมือนกัน ผู้เขียนโปรแกรมจะเลือกใช้งานรูปแบบใดก็ได้ แล้วแต่ความถนัดของแต่ละบุคคล ซึ่งลักษณะการเก็บข้อมูลของสตรัคเจอร์ทั้ง 2 รูปแบบเป็นดังนี้



การประกาศข้อมูลแบบสตรัคเจอร์ด้วยคีย์เวิร์ด typedef

typedef เป็นคีย์เวิร์ด ที่ใช้สำหรับกำหนดชนิดข้อมูลประเภทใหม่ขึ้นมาเอง เช่น กำหนด typedef int INTRGER; จะหมายถึง กำหนดให้ INTRGER แทนชนิดข้อมูลประเภท int ซึ่งสามารถนำ INTRGER ไปใช้กำหนดให้ตัวแปรต่างๆ มีชนิดข้อมูลเป็น int ได้

ตัวอย่างที่1 โปรแกรมแสดงการกำหนดชนิดข้อมูลประเภท

```
1   #include <stdio.h>
2   #include <conio.h>
3
4   main(){
5       typedef int INTEGER;
6       INTEGER a = 3 ;
7       printf("INTEGER a is %d\n", a);
8       getch();
9   }
```

ผลลัพธ์ของโปรแกรม

```
INTEGER a is 3
```

อธิบายโปรแกรม

บรรทัดที่5 : ใช้คีย์เวิร์ด typedef กำหนดชนิดข้อมูลประเภทใหม่ คือ INTEGER ขึ้น โดยให้มีความหมายเหมือนกับชนิดข้อมูลพื้นฐานประเภท int

บรรทัดที่6 : นำชนิดข้อมูลประเภทใหม่ คือ INTEGER กำหนดให้กับตัวแปร a ซึ่ง INTEGER a = 3 ; จะมีความหมายเหมือนกับการกำหนด int a=3;

```
typedef struct {
        ชนิดของข้อมูลตัวที่ 1  ตัวแปรตัวที่ 1;
        ชนิดของข้อมูลตัวที่ 2  ตัวแปรตัวที่ 2;
        ....
        ชนิดของข้อมูลตัวที่ n  ตัวแปรตัวที่ n;
} ชื่อที่ต้องการใช้กำหนดเป็นชนิดข้อมูลประเภทใหม่;
ชนิดข้อมูลประเภทใหม่ ชื่อตัวแปรที่ใช้อ้างอิงสตรัคเจอร์;
```

การประกาศข้อมูลแบบสตรัคเจอร์ด้วยคีย์เวิร์ด typedef และการประกาศข้อมูลแบบสตรัคเจอร์ ทั้ง 2 รูปแบบที่กล่าวไปแล้วข้างต้น สามารถใช้งานได้เหมือนกันทุกประการ เพียงแต่มีรูปแบบการประกาศข้อมูลที่แตกต่างกันเท่านั้น

การประกาศข้อมูลแบบสตรัคเจอร์ด้วยคีย์เวิร์ด typedef มีข้อที่ควรระวัง คือ

```
typedef struct{
    char name[10];
    int age;
}PERSON;
PERSON;
PERSON p;
```

1.ชื่อที่ประกาศตรงส่วนนี้ เป็นชื่อของชนิดข้อมูลประเภทใหม่ ไม่ใช่ชื่อตัวแปรที่ใช้อ้างอิงสตรัคเจอร์

2. การตั้งชื่อชนิดข้อมูลประเภทใหม่นี้นิยมใช้ตัวอักษรตัวใหญ่ทั้งหมด และมักใช้ _ (underscore) เป็นตัวคั่นในกรณีที่มีมากกว่า 1 คำ

การกำหนดค่าเริ่มต้นให้กับสมาชิกของสตรัคเจอร์

การกำหนดค่าเริ่มต้นให้กับสมาชิกของสตรัคเจอร์นั้น มีหลักการคล้ายกับการกำหนดค่าเริ่มต้นให้กับอาร์เรย์ คือ จะทำการกำหนดค่าเริ่มต้นไว้ภายในเครื่องหมาย { } และแยกค่าแต่ละค่าออกจากกันด้วยเครื่องหมาย , (comma) เช่น

```
struct person{
    char name[10];
    int age;
};
struct person ps = {"Ammie", 20};
```

ทำการกำหนดค่าเริ่มต้นให้กับสตรัคเจอร์ person โดยกำหนดให้สมาชิก name มีค่าเริ่มต้นเท่ากับ Ammie และกำหนดให้สมาชิก age มีค่าเริ่มต้น เท่ากับ 20

name	Ammie
age	20

```
typedef struct {
    char name[10];
    char sex ;
    char position[20];
    float salary;
}EMPLOYEE;
EMPLOYEE emp = {"Anan",'M',"programmer", 20000.00};
```

ทำการกำหนดชนิดข้อมูลประเภทใหม่ขึ้น คือ EMPLOYEE เพื่อใช้แทนสตรัคเจอร์ โดยกำหนดให้สมาชิก name มีค่าเป็น Ammie , sex เป็น M , position เป็น programmer , salary เป็น 20000.00

emp

name	Ammie
SEX	M
position	programmer
salary	20000.00

```
struct rentroom{
    int roomno;
    char owenrname[10];
    char surname[20];
    char sex ;
    int age;
    float prite;
}rr = {2019,"Ammie"};
```

ทำการกำหนดค่าเริ่มต้นให้กับสตรัคเจอร์ rentroom โดยกำหนดให้สมาชิก roomno มีค่าเป็น 2019 , ownername มีค่าเป็น Ammie ส่วนสมาชิกที่เหลือไม่ได้ทำการกำหนดค่าเริ่มต้นให้ ซึ่งหากสมาชิกตัวใดไม่ได้ถูกกำหนดค่าเริ่มต้นให้สมาชิกตัวนั้นจะถูกแทนที่ด้วยค่า null โดยถ้าสมาชิกตัวนั้นมีชนิดของข้อมูลเป็น int จะถูกแทนที่ด้วยค่า 0 แต่ถ้าเป็น float จะถูกแทนที่ด้วย 0.0 ส่วนถ้าสมาชิกมีชนิดข้อมูลเป็น char หรือ string จะถูกแทนที่ด้วยค่า \0

roomno	2019
ownername	Ammie
surname	\0
sex	\0
age	0
price	0.0

ลักษณะการจัดเก็บข้อมูลโครงสร้าง

เมื่อมีการประกาศใช้งานตัวแปรแบบโครงสร้าง คอมพิวเตอร์จะเตรียมเนื้อที่ภายในหน่วยความจำเพื่อจัดเก็บข้อมูลโครงสร้างที่ได้ประกาศขึ้น ซึ่งมีรูปแบบการจัดเก็บเป็นลำดับต่อเนื่อง

```
1   #include <stdio.h>
2
3   void main ()
4   {
5       struct members {
6           int id;
7           char name[20];
8           int age;
9       };
10
11      struct members info = {1,"Ammie",21};
12
13      printf ("\nAddress of id = %d", &info.id);
14      printf ("\nAddress of name = %d", &info.id);
15      printf ("\nAddress of age = %d", &info.id);
16      printf ("\n\n");
17  }
```

อธิบายโปรแกรม

บรรทัดที่5-9 : นิยามตัวแบบโครงสร้างชื่อ members ที่ประกอบด้วยสมาชิก id, name และage

บรรทัดที่11 : ประกาศให้ info เป็นตัวแปรโครงสร้างตามแบบโครงสร้างของ member พร้อมกำหนดค่าเริ่มต้น

บรรทัดที่13 : พิมพ์แอดเดรสของสมาชิก info.id

บรรทัดที่14 : พิมพ์แอดเดรสของสมาชิก info.name

บรรทัดที่15 : พิมพ์แอดเดรสของสมาชิก info.age

หน่วยการเรียนรู้ที่ ๑๐ การจัดการแฟ้มข้อมูลในภาษาซี

<u>มาตรฐานการเรียนรู้ / ตัวชี้วัด</u>

กลุ่มสาระการเรียนรู้วิทยาศาสตร์

สาระที่ ๔ เทคโนโลยี

มาตรฐาน ว ๔.๒ เข้าใจและใช้แนวคิดเชิงคำนวณในการแก้ปัญหาที่พบในชีวิตจริงอย่างเป็นขั้นตอนและเป็นระบบ ใช้เทคโนโลยีสารสนเทศและการสื่อสารในการเรียนรู้ การทำงาน และการแก้ปัญหาได้อย่างมีประสิทธิภาพ รู้เท่าทัน และมีจริยธรรม

<u>ตัวชี้วัด</u>

ว ๔.๒ ม.๑/๒ ออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อแก้ปัญหาทางคณิตศาสตร์หรือวิทยาศาสตร์

ว ๔.๒ ม.๒/๑ ออกแบบอัลกอริทึมที่ใช้แนวคิดเชิงคำนวณในการแก้ปัญหา หรือการทำงานที่พบในชีวิตจริง

ว ๔.๒ ม.๒/๒ ออกแบบและเขียนโปรแกรมที่ใช้ตรรกะและฟังก์ชันในการแก้ปัญหา

<u>สาระสำคัญ</u>

๑. เรียนรู้และทำความเข้าใจเกี่ยวกับโครงสร้างข้อมูลชนิดสตรัคเจอร์

๒. เรียนรู้และทำความเข้าใจเกี่ยวกับการประกาศข้อมูลแบบสตรัคเจอร์

<u>สาระการเรียนรู้</u>

- **ความรู้**
  - ๑. ความรู้เกี่ยวกับสตรัคเจอร์
  - ๒. ความรู้เกี่ยวกับสตริง
  - ๓. การประยุกต์เขียนโปรแกรมภาษาซี

- **ทักษะ / กระบวนการ**
  - ๑. แนะนำและทดลองใช้โปรแกรมพร้อมคำสั่งต่าง
  - ๒. อภิปราย และจำแนกรูปแบบต่าง ๆ ของเนื้อหา
  - ๓. ฝึกปฏิบัติเกี่ยวกับเนื้อหาทั้งหมด

- **คุณลักษณะที่พึงประสงค์**

๑. มีวินัย

๒. ใฝ่เรียนรู้

๓. มุ่งมั่นในการทำงาน

บทที่ 10 การจัดการแฟ้มข้อมูลในภาษาซี

การเขียนโปรแกรมที่ผ่านมา ล้วนเป็นการบันทึกข้อมูลลงในหน่วยความทรงจำหลัก ซึ่งเป็นหน่วยความจำแบบชั่วคราว โดยสังเกตได้จากข้อมูลที่เคยป้อน จะสูญหายไปทั้งหมดภายหลังจากจบโปรแกรมหรือปิดเครื่อง ทำให้เราสามารถอ้างอิงข้อมูลเดิมเพื่อใช้งานในครั้งหน้าได้อีก ดังนั้นหากต้องการเก็บข้อมูลไว้ใช้งาน จึงจำเป็นต้องบันทึกลงในอุปกรณ์สำรองข้อมูล เช่น ฮาร์ดดิสก์ ซึ่งเป็นอุปกรณ์จัดเก็บข้อมูลแบบถาวร ทำให้สามารถดึงข้อมูลที่เคยบันทึกไว้ นำออกมาใช้งานหรือปรับปรุงได้ตามต้องการ

ความหมายของแฟ้มข้อมูลในภาษา C

แฟ้มข้อมูล (data file) คือ แฟ้มที่มีการเก็บข้อมูลที่มีความสัมพันธ์กันมาไว้ด้วยกัน โดยมีการเก็บข้อมูลอย่างต่อเนื่องกันไป ตั้งแต่ต้นแฟ้มข้อมูลไปจนกระทั่งจบแฟ้มข้อมูล โดยที่ผู้เขียนข้อมูลสามารถแบ่งข้อมูลที่ต้องการจัดเก็บลงในแฟ้มเป็นฟิลด์ (field) หรือเรคอร์ด (record) ก็ได้ หรืออาจจัดเก็บข้อมูลตามแนวขนาดเนื้อที่โดยไม่จำเป็นต้องแบ่งข้อมูลในแฟ้มเป็นฟิลด์หรือเรคอร์ดก็ได้ โดยปกติผู้เขียนโปรแกรมภาษา C นิยมแบ่งข้อมูลที่ต้องการลงในแฟ้มเป็นฟิลด์หรือเรคอร์ด เพราะมีความสะดวกในการเรียกใช้ข้อมูลจากแฟ้มที่ต้องการนอกจากนี้ยังสามารถใช้โปรแกรม text editor หรือโปรแกรม word processing สร้างแฟ้มข้อมูลที่ต้องการได้อย่างสะดวกรวดเร็ว

ลำดับชั้นข้อมูล (Data Hierarchy)

การแบ่งลำดับชั้นของการจัดการข้อมูล (hierarchy of date)

ในการจัดการข้อมูลจะมีการแบ่งข้อมูลออกเป็นลำดับชั้น เพื่อง่ายต่อการเรียกใช้และประมวลผล

บิต (Bit = Binary Digit)

เป็นลำดับชั้นของหน่วยข้อมูลที่เล็กที่สุด ข้อมูลที่จะทำงานร่วมกับคอมพิวเตอร์ได้นั้น จะต้องเอามาแปลงให้อยู่ในรูปของเลขฐานสองก่อนคอมพิวเตอร์จึงจะเข้าใจและทำงานตามที่ต้องการได้ เมื่อแปลงแล้วจะได้ตัวเลขแทนสถานะเปิดและปิด ของสัญญาณไฟฟ้าที่เรียกว่า บิต เพียง 2 ค่า คือ บิต 0 และบิต 1

ไบต์ (Byte)

เมื่อนำบิตมารวมกันหลายๆบิต จะได้หน่วยข้อมูลกลุ่มใหม่ที่เรียกว่า ไบต์ (Byte) ซึ่งจำนวนของบิตที่ได้นั้นแต่ละกลุ่มอาจมีมากหรือน้อยบ้าง ทั้งนี้ขึ้นอยู่กับชนิดของรหัสที่ใช้เก็บ แต่โดยปกติกับการใช้งานในรหัสแอสกีทั่วไปจะได้กลุ่มของบิต 8 บิตด้วยกัน ซึ่งนิยมมาแทนเป็นรหัสของตัวอักษร บางครั้งจึงนิยมเรียกข้อมูล 1 ไบต์ว่าเป็น 1 ตัวอักษร

ฟิลด์ หรือเขตของข้อมูล (Field)

ประกอบด้วยกลุ่มของตัวอักษรหรือไบต์ตั้งแต่ 1 ตัวขึ้นไปมาประกอบกันเป็นหน่วยข้อมูลที่ใหญ่ขึ้น แล้วแสดงลักษณะหรือความหมายอย่างใดอย่างหนึ่ง ยกตัวอย่างเขตข้อมูลเกี่ยวกับพนักงาน เช่น รหัสพนักงาน ชื่อ นามสกุล เงินเดือน ตำแหน่ง

เรคอร์ด (Record)

เป็นกลุ่มของเขตข้อมูลหรือฟิลด์ที่มีความสัมพันธ์กัน และนำมาจัดเก็บรวมกันเป็นหน่วยใหม่ที่ใหญ่ขึ้นเพียงหน่วยเดียว ปกติในการจัดการข้อมูลมักประกอบด้วยเรคอร์ด ทั้งนี้ขึ้นอยู่กับขนาดของข้อมูลเป็นหลัก

ไฟล์ หรือแฟ้มตารางข้อมูล (File)

ไฟล์ หรือแฟ้มข้อมูล เป็นการนำเอาข้อมูลทั้งหมดหลายๆ เรคอร์ดที่ต้องการจัดเก็บมาเรียงอยู่ในรูปแบบของแฟ้มตารางข้อมูลเดียวกัน เช่น แฟ้มตารางข้อมูลเกี่ยวกับคะแนนนักศึกษาวิชาเทคโนโลยีสารสนเทศ อาจประกอบด้วยเรคอร์ดของนักศึกษาหลายๆคนที่เก็บข้อมูลเกี่ยวกับ รหัสนักศึกษา ชื่อ นามสกุล และคะแนนที่ได้

ประเภทของแฟ้มข้อมูล

ระบบแฟ้มข้อมูลในภาษา C มี 2 ประเภทด้วยกันคือ

1. เท็กซ์ไฟล์ (Text Files)

เป็นแฟ้มที่บรรจุไปด้วยตัวอักษรหรือข้อความที่จัดเรียงต่อกันเป็นลำดับ สำหรับข้อมูลที่บันทึกอยู่ในแฟ้มประเภทนี้ จะสามารถเปิดอ่านได้อย่างเข้าใจ เนื่องจากข้อมูลจะถูกจัดเก็บตามรหัสแอสกีของตัวอักขระนั้นๆ กรณีเท็กซ์ไฟล์ที่ใช้สำหรับบันทึกข้อมูลแบบเรคอร์ด แต่ละบรรทัดจะถูกแทนด้วยเรคอร์ดหนึ่งๆ และจะถูกปิดท้ายด้วยรหัส \n (newline) การขึ้นบรรทัดใหม่ ในภาษา C สามารถอ่าน หรือเขียนแฟ้มข้อมูลที่ละบรรทัด

2. ไบนารีไฟล์ (Binary Files)

เป็นแฟ้มที่ถูกจัดเก็บในรูปแบบรหัสไบนารีที่คอมพิวเตอร์สามารถอ่านและเข้าใจได้โดยตรง ที่สำคัญเราไม่สามารถสั่งพิมพ์ไบนารีไฟล์หรือเปิดอ่านได้โดยตรงเหมือนเท็กซ์ไฟล์ ข้อมูลที่จัดเก็บอยู่ในไบนารีไฟล์จะมีความกระทัดรัดมากกกว่า อีกทั้งยังสามารถเข้าถึงข้อมูลเพื่อปรับปรุงเปลี่ยนแปลงได้โดยตรง

ไบนารีไฟล์ เป็นแฟ้มข้อมูลที่ใช้เก็บเลขฐาน 2 ต่อเนื่องกัน จะไม่มีขอบเขตของบรรทัด เวลาใช้งานจึงต้องกำหนดขนาดในการอ่าน หรือเขียนเป็นจำนวนไบต์ (byte)

การเปิด / ปิด เท็กซ์ไฟล์

การทำงานร่วมกับไฟล์สิ่งแรกที่จะต้องทำคือเปิดไฟล์ขึ้นมาก่อน โดยการใช้ฟังก์ชัน fopen() ในการเปิดไฟล์ โดยมีรูปแบบการใช้งานดังนี้

```
FILE *fpt;
fpt = fopen(filename, mode);
```

โดยที่ : FILE *fpt; คือ การประกาศตัวแปร *fpt เป็นไฟล์พอยน์เตอร์ โดยตัวแปรชื่อ fpt นี้ สามารถตั้งชื่อด้วยชื่อใดก็ได้ ให้ตรงตามกฎเกณฑ์การตั้งชื่อในภาษา C แต่โดยส่วนใหญ่แล้วมักใช้คำว่า fpt (file pointer) เนื่องจากสื่อความหมายได้ดี ดังนั้นหากพบตัวแปร fpt ในโปรแกรมเมื่อใด ให้สันนิษฐานได้เลยว่า มีการกระทำกับไฟล์พอยน์เตอร์

fopen คือฟังก์ชันใช้สำหรับเปิดไฟล์

filename คือชื่อไฟล์

mode คือ โหมดการทำงานกับไฟล์ เช่น การสร้างไฟล์ หรือการอ่านไฟล์ข้อมูล

โหมด (mode) การทำงานของเท็กซ์ไฟล์แบ่งออกเป็น 6 ประเภท ดังนี้

โหมด	ความหมาย
r	เปิดไฟล์เพื่ออ่านข้อมูล - ถ้ามีไฟล์อยู่แล้ว จะทำการเปิดขึ้นมาเพื่ออ่าน - ถ้ายังไม่มีไฟล์ จะเกิด error
w	เปิดไฟล์เพื่อเขียนข้อมูล - ถ้ามีไฟล์อยู่แล้ว จะเขียนทับข้อมูลเดิมที่มีอยู่แล้วในไฟล์ - ถ้ายังไม่มีไฟล์ จะสร้างไฟล์ใหม่ขึ้นมาให้
a	เปิดไฟล์เพื่อเขียนข้อมูลต่อท้ายไฟล์เดิม - ถ้ามีไฟล์อยู่แล้ว จะเป็นการเขียนต่อท้ายจากไฟล์เดิม - ถ้ายังไม่มีไฟล์ จะสร้างไฟล์ใหม่ขึ้นมาให้
r+	เป็นการเปิดไฟล์ข้อมูลเก่า เพื่ออ่านหรือเขียนซ้ำทับไฟล์เก่า
w+	เป็นการเปิดไฟล์ข้อมูลเก่า เพื่ออ่านหรือเขียนข้อมูลทับไฟล์เก่า หรือสร้างไฟล์ใหม่
a+	เปิดไฟล์เก่าเพื่อเขียนข้อมูลต่อท้ายไฟล์เก่า หรือสร้างไฟล์ใหม่เพื่อเขียนข้อมูล

ตามปกติ เมื่อมีการเปิดไฟล์หรือแฟ้มข้อมูล ไฟล์พอยน์เตอร์จะถูกชี้ไปยังตำแหน่งแรกของไฟล์เสมอ ครั้นเมื่อมีคำสั่งประมวลผลแฟ้มเมื่อใด ไฟล์พอยน์เตอร์ก็จะเลื่อนไปยังตำแหน่งถัดไปเรื่อยๆ เพื่ออ่านข้อมูลเรคอร์ดถัดไป จนกระทั่งพบรหัส EOF (End of File) ที่หมายถึงจุดสิ้นสุดของไฟล์

ตัวอย่าง การเปิดไฟล์ด้วยโหมด “w” เพื่อสร้างเท็กซ์ไฟล์ใหม่

```
FILE *fpt;
fpt  =  fopen(“c:/myfile.txt” , “w”)
```

โดยที่

- กำหนดให้ตัวแปร fpt เป็นไฟล์พอยน์เตอร์

- กำหนดให้ fpt ซึ่งเป็นตัวแปรไฟล์พอยน์เตอร์ ชี้ไปยังตำแหน่งแรกของแฟ้มข้อมูลชื่อ myfile.txt

- โหมดการทำงานแบบ “w” ซึ่งหมายถึงการสร้างเท็กซ์ไฟล์ใหม่

สังเกตพาธที่กำหนดไว้คือ “c:/myfile.txt” ได้มีการใช้เครื่องหมาย / สาเหตุที่ไม่กำหนดเป็น “c:\myfile.txt” นั่นเป็นเพราะว่าในภาษา C นั้น หากมีการใช้เครื่องหมาย \ จะไปซ้ำกับรหัสควบคุม ดังนั้น ภาษา C จึงอนุโลมให้ใช้เครื่องหมาย / แทนเส้นทางของพาธได้ แต่ถ้าต้องการใช้เครื่องหมาย \ ก็สามารถทำได้เช่นกัน โดยใช้เครื่องหมาย \\ เช่น “c:\\myfile.txt” เมื่อไฟล์ถูกเปิดใช้งาน เสร็จแล้ว ก็ต้องสั่งปิดไฟล์ดังกล่าว

การปิดไฟล์

```
fclose(fpt);
```

โดยที่ fclose คือ ฟังก์ชันสำหรับสั่งปิดไฟล์

fpt คือ ตัวแปรไฟล์พอยน์เตอร์

ตัวอย่าง การปิดไฟล์

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
main() {
    FILE *fpt;
    fpt  =  fopen("c:/myfile.txt" , "w");  //open file
    if(fpt == NULL) {
        printf("Cannot open myfile.txt \n");
        exit(1);
    }
    fclose(fpt);      //close file
    getch();
}
```

อธิบายโปรแกรม

บรรทัดที่ 5 กำหนดตัวแปร fpt เป็นตัวแปรพอยน์เตอร์ที่มีชนิดข้อมูลเป็น File ขึ้นเพื่ออ้างอิงไฟล์

บรรทัดที่ 6 ใช้ฟังก์ชัน fopen() เปิดไฟล์ myfile.txt เพื่อเขียนข้อมูล

บรรทัดที่ 7-10 จากบรรทัดที่ 6 ได้นำตัวแปร fpt มารับค่าที่ฟังก์ชัน fopen() จะส่งคืนกลับมาให้ ดังนั้น บรรทัดที่ 7-10 จึงได้นำค่าของตัวแปร fpt มาทำการตรวจสอบว่าเปิดไฟล์สำเร็จหรือไม่

บรรทัดที่ 11 ใช้ฟังก์ชัน fclose() ทำการปิดไฟล์

หากตัวแปร fpt มีค่าเท่ากับ NULL แสดงว่าการเปิดไฟล์ไม่สำเร็จ ก็จะแสดงข้อความในบรรทัดที่ 8 เพื่อแจ้งให้ผู้เขียนโปรแกรมทราบว่าไม่สามารถเปิดไฟล์ได้ และบรรทัดที่ 9 ใช้คำสั่ง exit(1) สั่งให้ออกจากโปรแกรม โดยส่วนมากหากเป็นการจบการทำงานของโปรแกรมโดยสมบูรณ์แบบ คือไม่เกิด error แล้ว จะใช้คำสั่ง exit(0) ในการออกจากโปรแกรม แต่สำหรับกรณีเกิด error เราจะสั่งให้ออกจากการทำงานโดยใช้ exit (1) แทน กล่าวคือ exit(0) จะ return ค่า 0 หมายถึง success และ exit(1) จะ return ค่า 1 หมายถึง error นั่นเอง

การใช้ฟังก์ชัน exit() ต้องรวมเฮดเดอร์ไฟล์ stdlib.h ผนวกไว้ด้านบนด้วย

การปิดไฟล์ด้วยฟังก์ชัน fclose() นอกจากจะสั่งยกเลิกการใช้ไฟล์แล้ว ยังยืนยันการบันทึกข้อมูลลงในอุปกรณ์ด้วย อย่างไรก็ตาม หากต้องการสั่งให้โปรแกรมยืนยันการบันทึกข้อมูลลงในแฟ้ม โดยที่ยังคงเปิดไฟล์นั้นอยู่ ก็สามารถแทรกโค้ดคำสั่ง fflush() ลงในโปรแกรมก็ได้

```
fflush(fpt);
```

โดยที่ fflush คือ ฟังก์ชันยืนยันให้บันทึกข้อมูลลงในอุปกรณ์

fpt คือ ตัวแปรไฟล์พอยน์เตอร์

การบันทึกและอ่านข้อมูลจากเท็กซ์ไฟล์

ในไลบรารีมาตรฐาน (stdio.h) ได้จัดเตรียมฟังก์ชันต่างๆ สำหรับใช้บันทึกและอ่านข้อมูลจากไฟล์ ได้แก่

ฟังก์ชัน fputc() เป็นคำสั่งบันทึกข้อมูลทีละตัวอักษรลงในไฟล์

รูปแบบ

```
fputc(int c , fpt);
```

ฟังก์ชัน fgetc() เป็นคำสั่งอ่านข้อมูลทีละตัวอักษร

รูปแบบ

```
mchar = fgetc(fpt)
```

ฟังก์ชัน fputs() เป็นคำสั่งบันทึกข้อความสตริงทั้งบรรทัดจากไฟล์

รูปแบบ

```
fputs(char *str,fpt);
```

ฟังก์ชัน fgets()

รูปแบบ

```
fgets(char *str,int size,fpt);
```

ตัวอย่าง โปรแกรมการสร้างเท็กซ์ไฟล์ โดยรอรับค่าตัวอักขระทีละตัว แล้วบันทึกลงในไฟล์

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  main()
4  {
5      char ch;
6      FILE *fpt;
7      fpt  =  fopen("c:/c_code/sample.txt" , "w");
8      if(fpt == NULL) {
9          printf("Cannot open myfile.txt \n\n");
10         exit(0);
11     }
12     printf("Enter a character and press ENTER to exit:\n");
13     while(true)
14     {
15         ch = getchar();
16         if(ch == '\n')
17             break;
18         fputc(ch,fpt);
19     }
20     fclose(fpt);
21 }
```

คำอธิบายโปรแกรม

บรรทัดที่ 2 ผนวกเฮดเดอร์ไฟล์ stdlib.h เนื่องจากมีการใช้งาน exit() โดยเป็นคำสั่งให้โปรแกรมจบการทำงาน

บรรทัดที่ 5 กำหนดตัวแปร ch ให้มีชนิดข้อมูลเป็นตัวอักขระ 1 ตัว

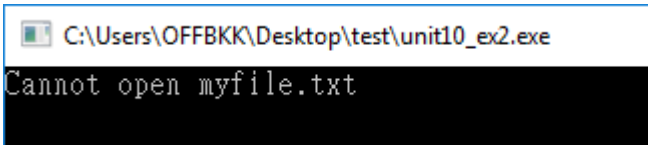
บรรทัดที่ 6-7 ประกาศตัวแปร fpt เป็นไฟล์พอยน์เตอร์ และเปิดไฟล์ใหม่ชื่อ sample.txt ภายใต้พาธ c:/c_code/ ด้วยโหมด “w” ซึ่งหมายถึงการสร้างไฟล์ใหม่

บรรทัดที่ 8-10 ตรวจสอบเงื่อนไขว่า เกิดข้อผิดพลาดจากการเปิดไฟล์หรือไม่ เช่นถ้าไม่พบพาธตำแหน่งไฟล์ที่ระบุไว้หรือดิสก์เต็ม ค่า fpt ก็จะมีค่าเท่ากับ NULL

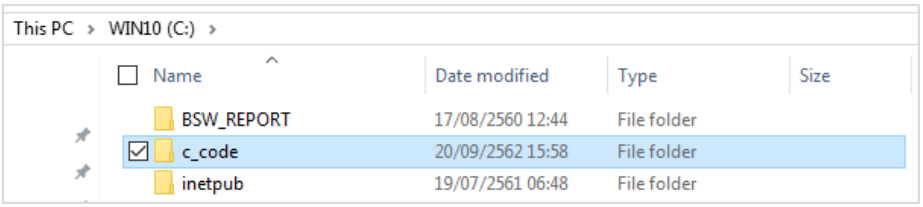
บรรทัดที่ 13-19 ลูปตัวอักขระทีละตัวจากแป้นพิมพ์ด้วยฟังก์ชัน getchar() โดยจะให้รอรับข้อมูลจากแป้นพิมพ์ไปจนกระทั่งมีการเคาะแป้น ENTER จึงหลุดออกจากลูป แต่ถ้ามีการกรอกอักขระใดๆลงไป ก็จะสั่งให้บันทึกตัวอักขระทีละตัวลงในไฟล์ด้วยฟังก์ชัน fputc()

บรรทัดที่ 20 ปิดไฟล์ข้อมูล

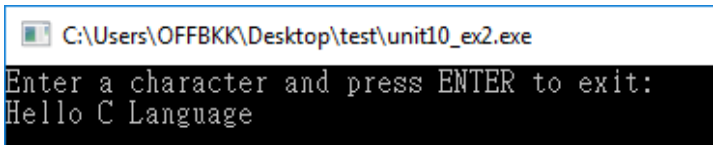
ผลลัพธ์โปรแกรม



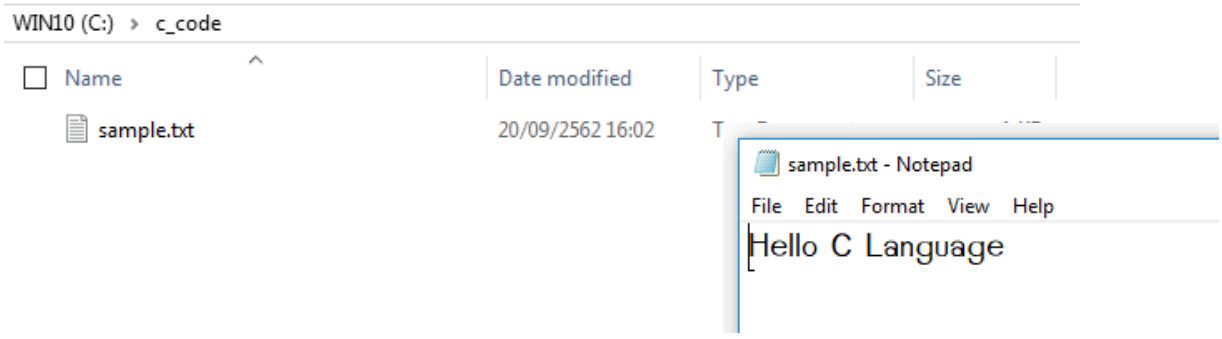
จากผลลัพธ์โปรแกรม จะพบว่าเกิดข้อผิดพลาดเกี่ยวกับไฟล์ขึ้น เนื่องจากไม่พบพาธตำแหน่งจัดเก็บไฟล์นั่นเอง ดังนั้นให้เปิดโปแกรม Explorer แล้วเข้าไปยังโฟลเดอร์ c:\c_code เพื่อสร้างโฟลเดอร์ sample ขึ้น



เมื่อสร้างโฟลเดอร์ c_code เป็นที่เรียบร้อยแล้ว ให้สั่งรันโปรแกรมอีกครั้ง แล้วกรอกข้อความลงไป



จากนั้นเปิดโปรแกรม Explorer ขึ้นมา แล้วเข้าไปยังโฟลเดอร์ c:\c_code จะพบเท็กซ์ไฟล์ชื่อ sample.txt ให้ดับเบิ้ลคลิกเพื่อเปิดไฟล์ดังกล่าวขึ้นมา



ตัวอย่าง โปรแกรมการอ่านเท็กซ์ไฟล์ ด้วยฟังก์ชัน fgetc()

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  main()
4  {
5      char ch;
6      FILE *fpt;
7      fpt  =  fopen("c:/c_code/sample.txt" , "r");
8      if(fpt == NULL)
9      {
10         printf("Error..cannot open file!! \n\n");
11         exit(0);
12     }
13     printf("Reading from a file and display : \n\n");
14     while(true)
15     {
16         ch = fgetc(fpt);
17         if(ch == EOF)
18             break;
19         else
20         printf("%c",ch);
21     }
22     fclose(fpt);
23 }
```

คำอธิบายโปรแกรม

บรรทัดที่ 2 ประกาศตัวแปร fpt เป็นไฟล์พอยน์เตอร์ และเปิดไฟล์ sample.txt ภายใต้พาธ c:/c_code/ ด้วยโหมด “r” ซึ่งหมายถึงการอ่านข้อมูลจากไฟล์

บรรทัดที่ 8-12 ตรวจสอบข้อผิดพลาดจากการเปิดไฟล์

บรรทัดที่ 14-21 สร้างลูป while เพื่ออ่านอักขระทีละตัวจากไฟล์ด้วยฟังก์ชัน fgetc() แล้วนำมาแสดงผลบนหน้าจอภาพโดยลูปนี้จะทำงานไปเรื่อยๆ จนกระทั่งจบไฟล์ ก็จะหลุดออกจากลูป

บรรทัดที่ 22 ปิดไฟล์

หน้าจอแสดงผลลัพธ์

```
C:\Users\OFFBKK\Desktop\test\unit10_ex3.exe
Reading from a file and display :

Hello C Language
```

จากตัวอย่างทั้ง 2 โปรแกรมที่ผ่านมา โปรแกรมการสร้างเท็กซ์ไฟล์ จะรอรับตัวอักขระทีละตัวจากแป้นพิมพ์ และบันทึกตัวอักขระแต่ละตัวลงในไฟล์ และโปรแกรมการอ่านเท็กซ์ไฟล์ จะอ่านอักขระทีละตัวจากไฟล์ และนำมาแสดงผลทางจอภาพ

ตัวอย่าง การสร้างเท็กซ์ไฟล์ เพื่อบันทึกข้อความสตริงลงในไฟล์ด้วยฟังก์ชัน fputs()

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #define SIZE 80
5  main()
6  {
7      char text[SIZE];
8      FILE *fpt;
9      fpt = fopen("c:/c_code/sample2.txt" , "w");
10     if(fpt == NULL)
11     {
12         printf("Error..cannot open file!! \n\n");
13         exit(0);
14     }
15     printf("Enter a few lines of text : \n\n");
16     while(strlen(gets(text)) > 0)
17     {
18         fputs(text,fpt);
19         fputs("\n",fpt);
20     }
21     fclose(fpt);
22 }
```

คำอธิบายโปรแกรม

บรรทัดที่ 4 ประกาศค่าคงที่ SIZE มีค่าเท่ากับ 80

บรรทัดที่ 7 ประกาศตัวแปร text เป็นอาร์เรย์เก็บข้อความ 80 ตัวอักษร
ประกาศตัวแปร fpt เป็นไฟล์พอยน์เตอร์

บรรทัดที่ 9 สร้างเท็กซ์ไฟล์ชื่อ sample2.txt ภายใต้พาธ c:/c_code/

บรรทัดที่ 10-14 ตรวจสอบเงื่อนไขว่า เกิดข้อผิดพลาดจากการเปิดไฟล์หรือไม่

บรรทัดที่ 16-20 ตรวจสอบเงื่อนไขว่า

ผลลัพธ์ของโปรแกรม

```
C:\Users\OFFBKK\Desktop\test\unit10_ex4.exe
Enter a few lines of text :

Pitchaporn Poonsawat
```