เอกสารประกอบการสอน

การโปรแกรมคอมพิวเตอร์

ผู้ช่วยศาสตราจารย์.ดร.พรภวิษย์ บุญศรีเมือง

คณะเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยราชภัฏสวนสุนันทา

๒๕๖๖

บทที่ 1

การวนซ้ำและการตัดสินใจ

1.1 บทนำ

ในการที่จะโปรแกรมสั่งการให้วนซ้ำและการตัดสินใจ (Loop and Decision) ที่รันคำสั่งทั้งหมดตามลำดับที่ตั้งแต่ต้นจนจบ โปรแกรมส่วนใหญ่จะตัดสินใจว่าจะทำอย่างไรเพื่อตอบสนองต่อสถานการณ์ที่เปลี่ยนแปลงไป โฟลว์ของการควบคุมจะข้ามจากส่วนหนึ่งของโปรแกรมไปยังอีกส่วนหนึ่ง ขึ้นอยู่กับการคำนวณที่ดำเนินการในโปรแกรม คำสั่งโปรแกรมที่ทำให้เกิดการกระโดดดังกล่าวเรียกว่าคำสั่งควบคุม มีสองประเภทหลัก: ลูปและการตัดสินใจจำนวนครั้งที่ดำเนินการวนซ้ำ หรือการตัดสินใจส่งผลให้เกิดการดำเนินการในส่วนของโค้ดหรือไม่ ขึ้นอยู่กับว่านิพจน์บางอย่างเป็นจริงหรือเท็จ โดยทั่วไปนิพจน์เหล่านี้เกี่ยวข้องกับตัวดำเนินการชนิดหนึ่งที่เรียกว่าตัวดำเนินการเชิงสัมพันธ์ ซึ่งเปรียบเทียบค่าสองค่า เนื่องจากการดำเนินการของลูปและการตัดสินใจเกี่ยวข้องอย่างใกล้ชิดกับตัวดำเนินการเหล่านี้ เราจะตรวจสอบก่อนตัวดำเนินการเชิงสัมพันธ์ ตัวดำเนินการเชิงสัมพันธ์จะเปรียบเทียบค่าสองค่า ค่าอาจเป็นประเภทข้อมูล C++ ในตัว เช่น char, int และ float หรือตามที่เราจะดูในภายหลัง อาจเป็นคลาสที่ผู้ใช้กำหนดก็ได้ การเปรียบเทียบเกี่ยวข้องกับความสัมพันธ์เช่น เท่ากับ น้อยกว่า และมากกว่า ผลลัพธ์ของการเปรียบเทียบเป็นจริงหรือเท็จ ตัวอย่างเช่น ค่าทั้งสองมีค่าเท่ากัน (จริง) หรือไม่ (เท็จ)
โปรแกรมแรกของเรา RELAT สาธิตตัวดำเนินการเชิงสัมพันธ์ในการเปรียบเทียบตัวแปรจำนวนเต็มและค่าคงที่

```
// relat.cpp
// demonstrates relational operators
#include <iostream>
using namespace std;

int main()

{

int numb;

  cout << “Enter a number: “;
  cin >> numb;
  cout << “numb<10  is “ << (numb < 10)  << endl;
```

```
cout << “numb>10  is “ << (numb > 10)  << endl;
cout << “numb==10 is “ << (numb == 10) << endl;
return 0;
}
```

ซึ่งค่าโปรแกรมนี้ทำการเปรียบเทียบสามประเภทระหว่าง 10 กับตัวเลขที่ผู้ใช้ป้อน นี่คือผลลัพธ์เมื่อผู้ใช้ป้อน 20:

```
Enter a number: 20
numb<10  is 0
numb>10  is 1
numb==10 is 0
```

นิพจน์แรกเป็นจริงหาก numb น้อยกว่า 10 นิพจน์ที่สองเป็นจริงหาก numb มากกว่า 10 และนิพจน์ที่สามเป็นจริงหาก numb เท่ากับ 10 ดังที่เห็นจากเอาต์พุต คอมไพเลอร์ C++ จะพิจารณาว่า นิพจน์จริงมีค่า 1 ในขณะที่นิพจน์เท็จมีค่า 0
ดังที่เราได้กล่าวไว้ในบทที่แล้ว Standard C++ มีประเภทบูลซึ่งสามารถเก็บค่าคงที่ค่าใดค่าหนึ่งจากสองค่า จริงหรือเท็จ คุณอาจคิดว่าผลลัพธ์ของนิพจน์เชิงสัมพันธ์ เช่น numb<10 จะเป็นประเภท bool และโปรแกรมจะพิมพ์ false แทนที่จะเป็น 0 และ true แทนที่จะเป็น 1 อันที่จริง C++ ค่อนข้างจะเป็นโรคจิตเภทในประเด็นนี้ การแสดงผลลัพธ์ของการดำเนินการเชิงสัมพันธ์ หรือแม้แต่ค่าของตัวแปรประเภทบูล ด้วย cout<< ให้ผลเป็น 0 หรือ 1 ไม่ใช่เท็จหรือจริง ในอดีตนี่เป็นเพราะว่า C++ เริ่มต้นโดยไม่มีประเภทบูล ก่อนการถือกำเนิดของ Standard C++ วิธีเดียวที่จะแสดงค่า false และ true ได้คือ 0 และ 1 ขณะนี้ false สามารถแสดงด้วยค่าบูลเป็น false หรือด้วยค่าจำนวนเต็มเป็น 0; และจริงสามารถแสดงด้วยค่าบูลที่เป็นจริงหรือค่าจำนวนเต็มเป็น 1
ในสถานการณ์ทั่วไปส่วนใหญ่ ความแตกต่างจะไม่ปรากฏชัดเจนเนื่องจากเราไม่จำเป็นต้องแสดงค่าจริง/เท็จ เราแค่ใช้มันในลูปและการตัดสินใจเพื่อกำหนดสิ่งที่โปรแกรมจะทำต่อไป

Operator Meaning

> Greater than (greater than) < Less than
== Equal to
!= Not equal to

>= Greater than or equal to <= Less than or equal to

ลูป (Loops)

การวนซ้ำจะทำให้ส่วนของโปรแกรมของคุณถูกทำซ้ำตามจำนวนครั้งที่กำหนด การทำซ้ำจะดำเนินต่อไปในขณะที่เงื่อนไขเป็นจริง เมื่อเงื่อนไขกลายเป็นเท็จ การวนซ้ำจะสิ้นสุดและการควบคุมจะส่งผ่านไปยังคำสั่งที่ตามหลังการวนซ้ำ
ลูปในภาษา C++ มีสามประเภท ได้แก่ ลูป for, ลูป while และลูป do

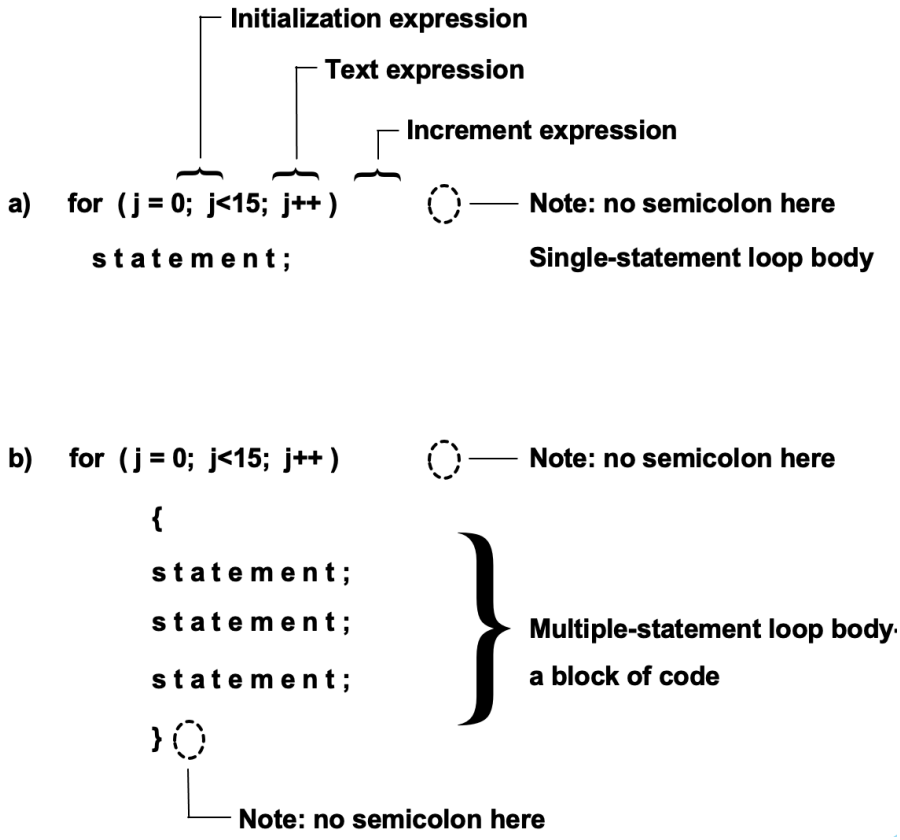
การวนซ้ำ (for Loop)

for loop นั้นเป็นลูป C++ ที่ง่ายที่สุดในการเข้าใจ (สำหรับหลาย ๆ คน) องค์ประกอบการควบคุมลูปทั้งหมดถูกรวบรวมไว้ในที่เดียว ในขณะที่โครงสร้างลูปอื่นๆ จะกระจัดกระจายเกี่ยวกับโปรแกรม ซึ่งสามารถทำให้มันยากขึ้นในการคลี่คลายว่าลูปเหล่านี้ทำงานอย่างไร for loop ดำเนินการส่วนของโค้ดตามจำนวนครั้งที่คงที่ โดยปกติ (แม้ว่าจะไม่เสมอไป) จะใช้เมื่อคุณทราบก่อนจะเข้าสู่ลูปว่าต้องการรันโค้ดกี่ครั้งต่อไปนี้คือตัวอย่าง FORDEMO ที่แสดงกำลังสองของตัวเลขตั้งแต่ 0 ถึง 14:

```
// fordemo.cpp
// demonstrates simple FOR loop
#include <iostream>
using namespace std;

int main() {

int j;

  for(j=0; j<15; j++)
    cout << j * j << “  “;  //displaying the square of j
  cout << endl;
  return 0;
  }
```

ผลลัพธ์จะได้
0 1 4 9 16 25 36 49 64 81 100 121 144 169 196

การทำงานอย่างไร? คำสั่ง for ควบคุมการวนซ้ำ ประกอบด้วยคำสำคัญสำหรับ, โฟล-ต่ำสุดด้วยวงเล็บที่มีสามนิพจน์คั่นด้วยเครื่องหมายอัฒภาค สำหรับ(j=0; j<15; j++) นิพจน์ทั้งสามนี้คือ นิพจน์การเริ่มต้น นิพจน์ทดสอบ และนิพจน์ส่วนเพิ่ม ดังแสดงในรูปที่ 1



รูปที่ 1 แสดง Syntax of the for Loop

นิพจน์ทั้งสามนี้มักจะเกี่ยวข้องกับตัวแปรเดียวกัน ซึ่งเราเรียกว่าตัวแปรลูป ในตัวอย่าง FORDEMO ตัวแปรลูปคือ j ถูกกำหนดไว้ก่อนที่คำสั่งภายในเนื้อหาลูปจะเริ่มดำเนินการเนื้อความของลูปคือโค้ดที่จะดำเนินการในแต่ละครั้งผ่านการวนซ้ำ การทำซ้ำโค้ดนี้ถือเป็นผลลัพธ์สำหรับการวนซ้ำ ในตัวอย่างนี้เนื้อความของวงประกอบด้วยคำสั่งเดียว

```
cout << j * j << " ";
```

คำสั่งจะพิมพ์กำลังสองของ j ตามด้วยช่องว่างสองช่อง สี่เหลี่ยมจัตุรัสหาได้จากการนำ j คูณด้วยตัวมันเอง ในขณะที่ลูปดำเนินการ j จะผ่านลำดับ 0, 1, 2, 3 และอื่น ๆ จนถึง 14; ดังนั้นกำลังสองของตัวเลขเหล่านี้จึงแสดงขึ้น ได้แก่ 0, 1, 4, 9, สูงสุด 196 และคำสั่ง for จะไม่ตามด้วยเครื่องหมายอัฒภาค นั่นเป็นเพราะว่าคำสั่ง for และเนื้อหาวนซ้ำรวมกันถือเป็นคำสั่งโปรแกรม นี่เป็นรายละเอียดที่สำคัญ หากคุณใส่เครื่องหมายอัฒภาคไว้หลังคำสั่ง for คอมไพลเลอร์จะคิดว่าไม่มีเนื้อหาแบบวนซ้ำ และสามนิพจน์ในคำสั่ง for ควบคุมลูปได้อย่างไร

นิพจน์การเริ่มต้น

นิพจน์การเริ่มต้นจะถูกดำเนินการเพียงครั้งเดียว เมื่อการวนซ้ำเริ่มต้นครั้งแรก มันให้ค่าเริ่มต้นแก่ตัวแปรลูป ในตัวอย่าง FORDEMO จะตั้งค่า j เป็น 0

นิพจน์ทดสอบ

นิพจน์การทดสอบมักจะเกี่ยวข้องกับตัวดำเนินการเชิงสัมพันธ์ จะถูกประเมินแต่ละครั้งผ่านการวนซ้ำ ก่อนที่เนื้อความของลูปจะถูกดำเนินการ มันกำหนดว่าการวนซ้ำจะถูกดำเนินการอีกครั้งหรือไม่ หากนิพจน์ทดสอบเป็นจริง การวนซ้ำจะถูกดำเนินการอีกครั้ง หากเป็นเท็จ การวนซ้ำจะสิ้นสุดและการควบคุมจะส่งผ่านไปยังคำสั่งที่ตามหลังการวนซ้ำ ในตัวอย่าง FORDEMO คำสั่ง

cout<< endl;

จะถูกดำเนินการหลังจากเสร็จสิ้นการวนซ้ำ

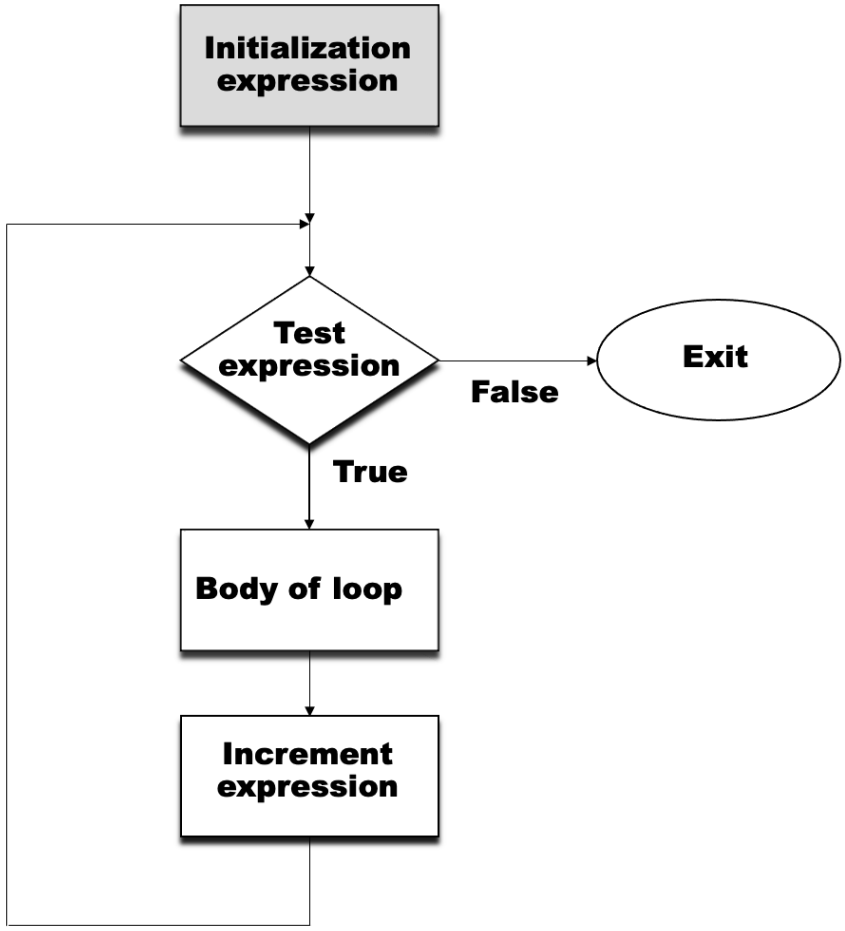
นิพจน์ที่เพิ่มขึ้น

นิพจน์ส่วนเพิ่มจะเปลี่ยนค่าของตัวแปรลูป บ่อยครั้งโดยการเพิ่มค่า มันจะถูกดำเนินการที่จุดสิ้นสุดของลูปเสมอ หลังจากที่ตัวเนื้อหาของลูปถูกดำเนินการแล้ว ที่นี่ตัวดำเนินการเพิ่ม ++ เพิ่ม 1 ถึง j แต่ละครั้งผ่านการวนซ้ำ รูปที่ 3.2 แสดงผังงานของการดำเนินการของ for loop

กี่ครั้งของการวนซ้ำ

การวนซ้ำในตัวอย่าง FORDEMO ดำเนินการ 15 ครั้งพอดี ครั้งแรก j คือ 0 ซึ่งมั่นใจได้ในนิพจน์การเริ่มต้น ครั้งสุดท้ายที่ผ่านลูป j คือ 14 ซึ่งถูกกำหนดโดยนิพจน์ทดสอบ j<15 เมื่อ j กลายเป็น 15 การวนซ้ำจะสิ้นสุดลง เนื้อความของลูปจะไม่ถูกดำเนินการเมื่อ j มีค่านี้ การจัดเรียงที่แสดงนี้มักใช้

เพื่อทำบางสิ่งในจำนวนครั้งคงที่: เริ่มต้นที่ 0 ใช้นิพจน์ทดสอบกับตัวดำเนินการน้อยกว่าและค่าเท่ากับจำนวนการวนซ้ำที่ต้องการ และเพิ่มตัวแปรลูปหลังการวนซ้ำแต่ละครั้ง - เหตุผล



รูปที่ 2 Operation of the **for LooP**

คำสั่งหลายรายการในเนื้อความของลูป (Multiple Statements in the Loop Body)

การดำเนินการมากกว่าหนึ่งคำสั่งในเนื้อหาของลูป คำสั่งหลายคำสั่งคั่นด้วยเครื่องหมายปีกกา เช่นเดียวกับฟังก์ชัน โปรดทราบว่าไม่มีอัฒภาคตามวงเล็บปีกกาสุดท้ายของตัวลูป แม้ว่าจะมี

อัฒภาคตามหลังคำสั่งแต่ละตัวในตัวลูปก็ตาม ตัวอย่างถัดไป CUBELIST ใช้สามคำสั่งในส่วนเนื้อหาของลูป โดยจะพิมพ์ลูกบาศก์ของตัวเลขตั้งแต่ 1 ถึง 10 โดยใช้รูปแบบสองคอลัมน์

```
// cubelist.cpp
// lists cubes from 1 to 10
#include <iostream>
#include <iomanip>
using namespace std;

int main() {

int numb;

  for(numb=1; numb<=10; numb++)
    {
    cout << setw(4) << numb;
    int cube = numb*numb*numb;
    cout << setw(6) << cube << endl;  //display 2nd column
    }

return 0; }
```

ซึ่งจะได้ผลลัพธ์ของโปรแกรมดังต่อไปนี้

```
1     1
2     8
 3     27
 4     64
5     125
6     216
```

7 343

8 512

9 729

10 1000

การดีบักแอนิเมชั่น (Animation)

คุณสามารถใช้คุณสมบัติการดีบักที่สร้างไว้ในคอมไพเลอร์เพื่อสร้างการแสดงภาพเคลื่อนไหวที่การดำเนินการวนซ้ำ คุณสมบัติที่สำคัญคือก้าวเดียว คอมไพเลอร์ทำให้สิ่งนี้ง่ายขึ้น เริ่มต้นด้วยการเปิดโปรเจ็กต์เพื่อให้โปรแกรมทำการดีบั๊ก และหน้าต่างที่มีไฟล์ต้นฉบับ คำแนะนำที่แน่นอนที่จำเป็นในการเปิดใช้ดีบักเกอร์จะแตกต่างกันไปตามคอมไพเลอร์ที่แตกต่างกัน ดังนั้นโปรดดู Appendix C, “Microsoft Visual C++” หรือ Appendix D, “Borland C++Builder” ตามความเหมาะสม ด้วยการกดปุ่มฟังก์ชันบางปุ่มสามารถทำให้โปรแกรมของคุณรันได้ครั้งละหนึ่งบรรทัด นี่จะแสดงให้เห็นลำดับของคำสั่งที่ดำเนินการในขณะที่โปรแกรมดำเนินการ ในลูปจะเห็นคำสั่งภายในลูปที่ดำเนินการ จากนั้นการควบคุมจะกระโดดกลับไปที่จุดเริ่มต้นของลูปและวงจรจะทำซ้ำและยังสามารถใช้ดีบักเกอร์เพื่อดูว่าเกิดอะไรขึ้นกับค่าของตัวแปรต่างๆ เมื่อผ่านขั้นตอนเดียวของโปรแกรม นี่เป็นเครื่องมือที่ดีเมื่อทำการดีบั๊กโปรแกรมสามารถทดลองเทคนิคนี้กับโปรแกรม CUBELIST ได้โดยใส่ตัวแปรชาและคิวบ์ในหน้าต่าง Watch ในดีบักเกอร์และดูว่าตัวแปรเหล่านั้นเปลี่ยนแปลงไปอย่างไรเมื่อโปรแกรมดำเนินไป Single-stepping และหน้าต่าง Watch เป็นเครื่องมือแก้ไขจุดบกพร่องที่มีประสิทธิภาพ

for Loop Variations สำหรับรูปแบบลูป

นิพจน์ส่วนเพิ่มไม่จำเป็นต้องเพิ่มตัวแปรลูป มันสามารถดำเนินการใดๆ ในตัวอย่างถัดไปมันจะลดตัวแปรลูป โปรแกรม FACTOR นี้ขอให้ผู้ใช้พิมพ์ตัวเลขแล้วคำนวณแฟกทอเรียลของตัวเลขนี้ (แฟกทอเรียลคำนวณโดยการคูณจำนวนเดิมด้วยจำนวนเต็มบวกทั้งหมดที่น้อยกว่าตัวมันเอง ดังนั้นแฟกทอเรียลของ 5 คือ 5*4*3*2*1 หรือ 120)

```
// factor.cpp
// calculates factorials, demonstrates FOR loop
#include <iostream>
using namespace std;
```

```
int main() {

   unsigned int numb;
   unsigned long fact=1;
   cout << “Enter a number: “;
   cin >> numb;
//long for larger numbers

//get number

   for(int j=numb; j>0; j--)        //multiply 1 by
      fact *= j;                    //numb, numb-1, ..., 2, 1
   cout << “Factorial is “ << fact << endl;
   return 0;
   }
```

ในตัวอย่างนี้ นิพจน์การเริ่มต้นจะตั้งค่า j เป็นค่าที่ผู้ใช้ป้อน นิพจน์ทดสอบทำให้ลูปดำเนินการตราบใดที่ j มากกว่า 0 นิพจน์ที่เพิ่มขึ้นจะลดลง j หลังจากการวนซ้ำแต่ละครั้งในประเภท unsigned long สำหรับแฟกทอเรียล เนื่องจากแฟกทอเรียลของจำนวนที่น้อยมากก็มีขนาดใหญ่มากบนระบบ 32 บิต เช่น Windows int นั้นเหมือนกับ long แต่ long จะเพิ่มความจุให้กับระบบ 16 บิต ผลลัพธ์ต่อไปนี้แสดงให้เห็นว่าแฟกทอเรียลสามารถมีได้มากเพียงใด แม้ว่าตัวเลขอินพุตจะน้อยก็ตาม:

ใส่ตัวเลข: 10

แฟกทอเรียลคือ 3628800 จำนวนที่มากที่สุดที่คุณสามารถใช้สำหรับอินพุตคือ 12 คุณจะไม่ได้รับข้อความแสดงข้อผิดพลาดสำหรับอินพุตที่มากขึ้น แต่ผลลัพธ์จะผิด เนื่องจากความจุของประเภทยาวจะเกิน

While Loop

for loop ทำบางสิ่งในจำนวนครั้งคงที่ จะเกิดอะไรขึ้นหากไม่รู้ว่าต้องการทำอะไรสักกี่ครั้งก่อนที่จะเริ่มวนซ้ำ ในกรณีนี้อาจใช้การวนซ้ำประเภทอื่น: การวนซ้ำ while

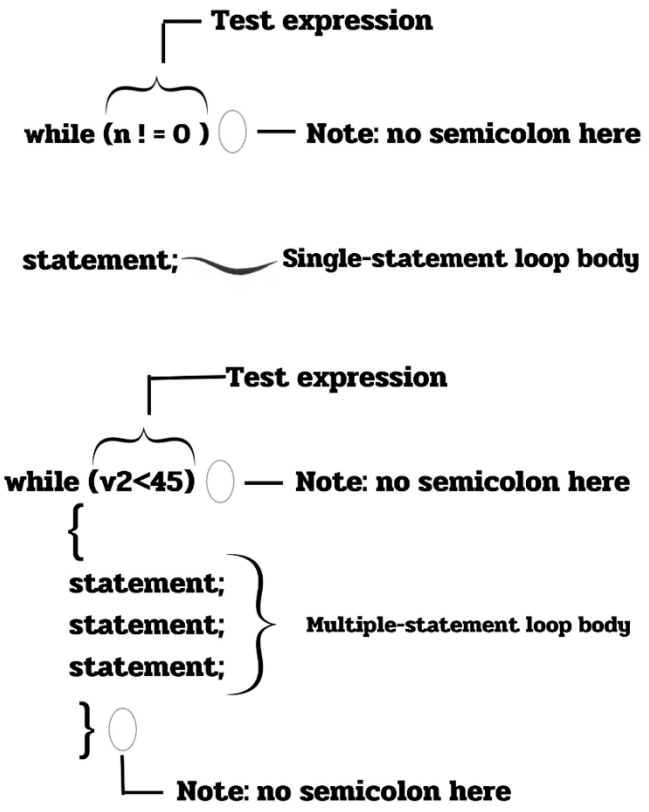
ตัวอย่างถัดไป ENDON0 ขอให้ผู้ใช้ป้อนชุดตัวเลข เมื่อตัวเลขที่ป้อนเป็น 0 การวนซ้ำจะสิ้นสุดลง โปรดสังเกตว่าไม่มีทางที่โปรแกรมจะรู้ล่วงหน้าได้ว่าจะต้องพิมพ์ตัวเลขจำนวนเท่าใดก่อนที่เลข 0 จะปรากฏ นั่นขึ้นอยู่กับผู้ใช้

```
// endon0.cpp
// demonstrates WHILE loop
#include <iostream>
using namespace std;

int main() {

int n = 99;

  while( n != 0 )
    cin >> n;
  cout << endl;
  return 0;
  }
```

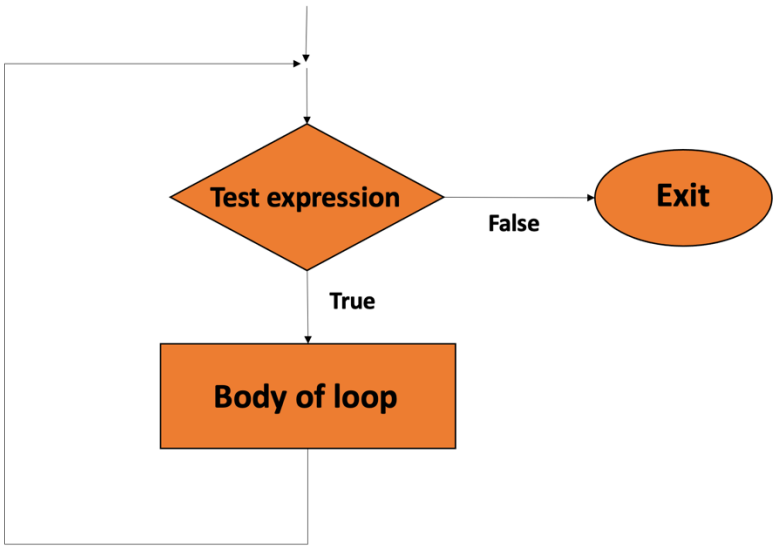
นี่คือตัวอย่างผลลัพธ์บางส่วน ผู้ใช้ป้อนตัวเลข และการวนซ้ำจะดำเนินต่อไปจนกระทั่งป้อน 0 ซึ่งเป็นจุดที่การวนซ้ำและโปรแกรมยุติลง

```
1

27

33

144

 9

0
```

while loop ดูเหมือน for for loop เวอร์ชันที่ง่าย ประกอบด้วยนิพจน์ทดสอบ แต่ไม่มีนิพจน์การเริ่มต้นหรือส่วนเพิ่ม รูปที่ 3 แสดงไวยากรณ์ของลูป while



รูปที่ 3 syntax of the While loop



รูปที่ 4 Operation of the while loop

```
// while4.cpp
// prints numbers raised to fourth power
#include <iostream>
#include <iomanip>
using namespace std;

int main() {

  int pow=1;
  int numb=1;
  while( pow<10000 )
    {
    cout << setw(2) << numb;
    cout << setw(5) << pow << endl;  //display fourth power
  ++numb;
  pow = numb*numb*numb*numb;
  }
```

```
cout << endl;
return 0;
}
//get ready for next power
//calculate fourth power
```

เมื่อต้องการหาค่าแค่คูณมันด้วยตัวมันเองสี่ครั้ง แต่ละครั้งที่วนซ้ำจะงงมากขึ้น แต่เราไม่ได้ใช้ numb ในนิพจน์ทดสอบในขณะที่ แต่ค่าผลลัพธ์ของ pow จะเป็นตัวกำหนดว่าจะยุติการวนซ้ำเมื่อใด

Statement

คำสั่ง if เป็นคำสั่งที่ง่ายที่สุดในการตัดสินใจ

```
// ifdemo.cpp
// demonstrates IF statement
#include <iostream>
using namespace std;

int main() {

int x;

  cout << “Enter a number: “;
  cin >> x;
  if( x > 100 )
    cout << “That number is greater than 100\n”;
  return 0;

}
```

Multiple Statements in the if Body

```
// if2.cpp
// demonstrates IF with multiline body
```

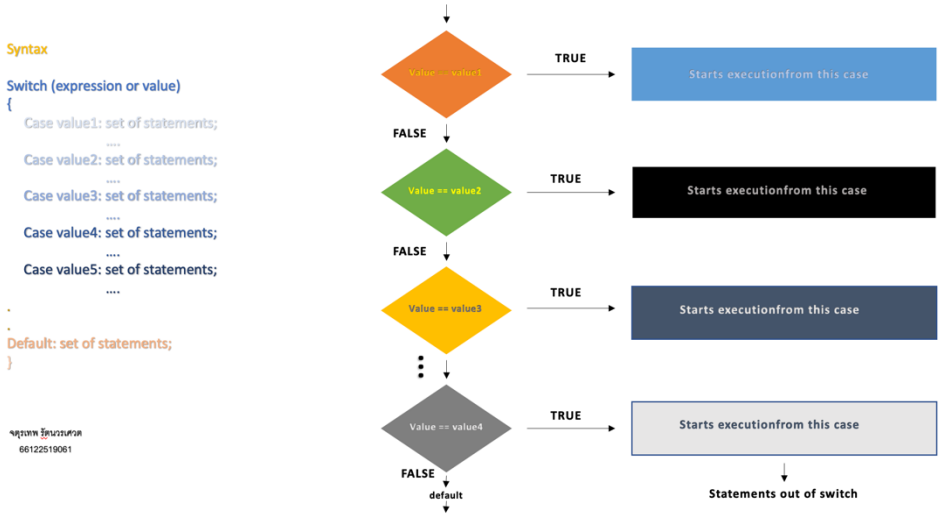
```
#include <iostream>
using namespace std;

int main() {

int x;

  cout << “Enter a number: “;
  cin >> x;
  if( x > 100 )
    {
    cout << “The number “ << x;
    cout << “ is greater than 100\n”;
    }

return 0; } =
```

Nesting ifs Inside Loops

โครงสร้างการวนซ้ำและการตัดสินใจที่เราเคยเห็นมาสามารถซ้อนกันได้ คุณสามารถซ้อน ifs ภายในลูป, วนซ้ำภายใน ifs, ifs ภายใน ifs และอื่นๆ นี่คือตัวอย่าง PRIME ที่ซ้อน if ไว้ใน for loop ตัวอย่างนี้จะบอกว่าตัวเลขที่เราป้อนเป็นจำนวนเฉพาะหรือไม่และจำนวนเฉพาะเป็นจำนวนเต็มหารด้วยตัวมันเองและ 1 เท่านั้น โดยจำนวนเฉพาะสองสามตัวแรกได้แก่ 2, 3, 5, 7, 11, 13, 17

```
// prime.cpp
// demonstrates IF statement with prime numbers
#include <iostream>
using namespace std;
#include <process.h>

int main() {

  unsigned long n, j;

//for exit()

cout << “Enter a number: “;
```

```
cin >> n;
for(j=2; j <= n/2; j++)

if(n%j == 0) {

//get number to test
//divide by every integer from
//2 on up; if remainder is 0,
//it's divisible by j
        cout << "It's not prime; divisible by " << j << endl;
        exit(0);                //exit from the program
        }
   cout << "It's prime\n";
   return 0;
   }
```



รูปที่ 5 Operation of the switch statement

หนังสืออ้างอิง

[1] Ulla Kirch-Prinz ,Peter Prinz , “A Complete Guide to Programming in C++” 4th Sams Publishing 2002

[2] Learn C++ programming language tutorialspoint www.tutorialspoint.com