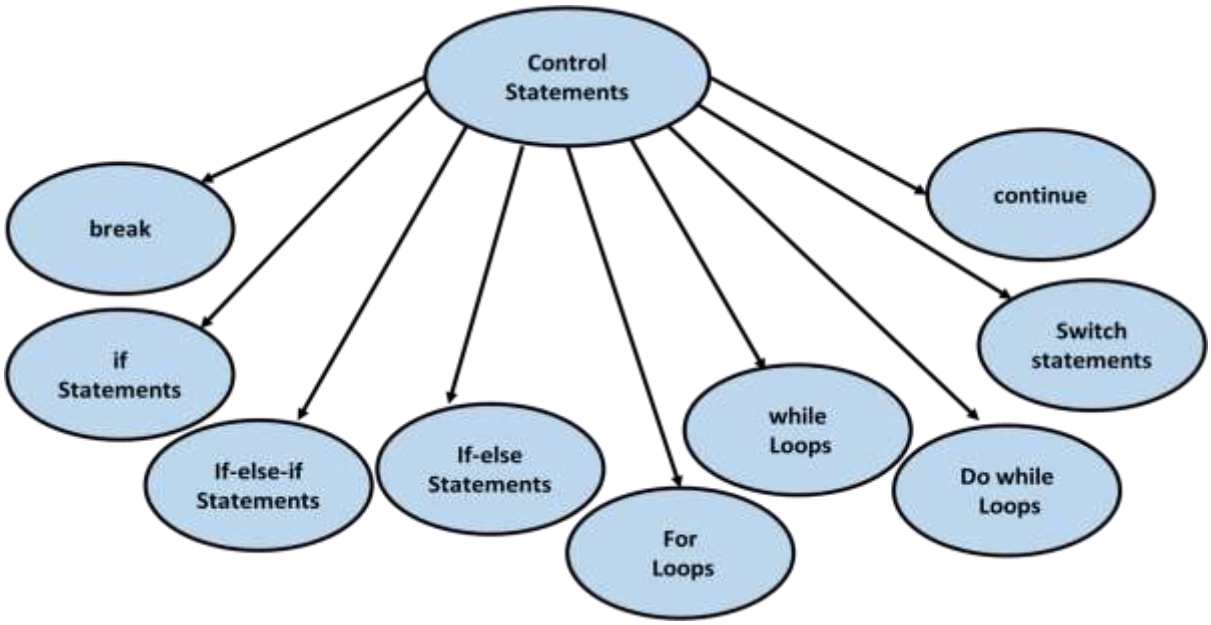
หน่วยที่ 2

คำสั่งควบคุม (Control Statement)



หัวข้อเรื่อง

2.1 คำสั่งแบบทางเลือก (selection Statement)

2.2 คำสั่งแบบการทำซ้ำ (repetition Statement)

2.3 การใช้คำสั่ง break, continue และ exit(0)

2. คำสั่งควบคุม (Control Statement)

โดยปกติ คำสั่งในโปรแกรมจะทำงานต่อเนื่องกันไปตามลำดับที่เขียนไว้ในโปรแกรม โดยจะเรียกการทำงานอย่างนี้ว่า sequential execution (การทำงานตามลำดับ) เมื่อมีการเขียนโปรแกรมที่ซับซ้อนหรือมีเงื่อนไขการทำงานมาเกี่ยวข้อง การเขียนโปรแกรมจึงต้องใช้คำสั่งการควบคุมว่าคำสั่ง ถัดไปที่จะทำงานเป็นคำสั่งใด ซึ่งเรียกสิ่งนี้ว่า Transfer of control (การส่งถ่ายการควบคุม) ซึ่งการส่งถ่ายการควบคุมที่ดีเป็นสิ่งสำคัญในการพัฒนาโปรแกรมอย่างมาก Bohm และ Jacopini ได้แสดงให้เห็นว่าโปรแกรมทุกโปรแกรมสามารถเขียนขึ้นได้โดยใช้คำสั่งมาทำการควบคุมเพียง 3 ชนิดเท่านั้น คือ

- คำสั่งตามลำดับ (sequence Statement)
- คำสั่งทางเลือก (selection Statement)
- คำสั่งการทำซ้ำ (repetition Statement)

โดยคำสั่งตามลำดับเป็นพื้นฐานทั่วไปในการเขียนคำสั่ง สำหรับโปรแกรมที่มีเงื่อนไขจะต้องใช้การเขียนโปรแกรมแบบทางเลือกหรือแบบการทำซ้ำ ซึ่งใช้ศึกษาในหน่วยนี้

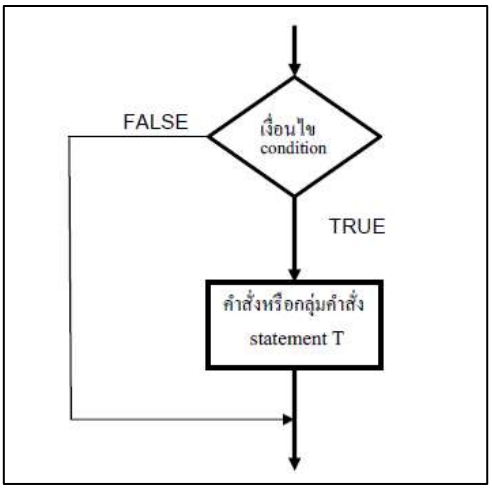
2.1 คำสั่งแบบทางเลือก (selection Statement)

คำสั่งแบบทางเลือก if คือ คำสั่งสำหรับใช้เลือกทำคำสั่ง (หรือกลุ่มคำสั่ง) โดยอาศัยการตรวจสอบเงื่อนไข ซึ่งมี 3 รูปแบบคือ

- หนึ่งทางเลือก (One Alternative)
- สองทางเลือก (Two Alternative)
- หลายทางเลือก (Multiple-Alternative)

2.1.1 หนึ่งทางเลือก (One Alternative)

คำสั่งแบบทางเลือก if ในรูปแบบนี้จะเลือกทำหนึ่งคำสั่ง (หรือกลุ่มคำสั่ง) ก็ต่อเมื่อตรวจสอบเงื่อนไขแล้วเป็นจริง คำสั่งแบบนี้สามารถเขียน flowchart ได้ตามรูป



ภาพที่ 1 แสดง flowchart

รูปแบบของคำสั่ง if

รูปแบบการทำ 1 คำสั่ง : if (condition)

Statement T;

รูปแบบการทำ กลุ่มคำสั่ง : if (condition)

{

Statement T;

Statement T2;

}

ตัวอย่างที่ 2.1 จงเขียนโปรแกรมรับข้อมูลเลขจำนวนเต็มทางแป้นพิมพ์ แล้วทำการเปรียบเทียบกับเลขจำนวนเต็ม100 ถ้าตัวเลขที่รับเข้ามานั้นมากกว่าให้แสดงข้อความ That number is greater than 100 ทางจอภาพ

```
#include <iostream>
using namespace std;
int main()
{
	int x;
	cout << "Enter integer number: ";
	cin >> x;
	if( x > 100 )
	cout << "That number is greater than 100\n";
	return 0;
}
```

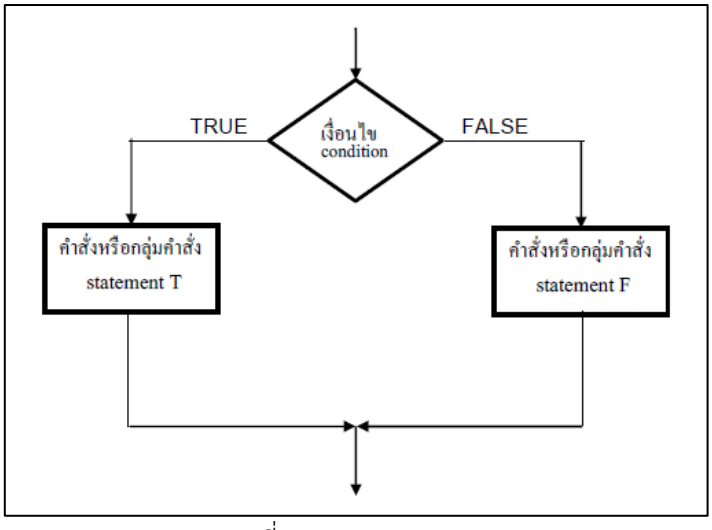
ผลลัพธ์

Enter a number: 200

That number is greater than 100

2.1.2 สองทางเลือก (Two Alternative)

คำสั่งแบบทางเลือก if ในรูปแบบนี้มีสองทางเลือก กล่าวคือ เลือกทำหนึ่งคำสั่ง (หรือกลุ่มคำสั่ง เมื่อตรวจสอบเงื่อนไขแล้วเป็นจริง หรือ ทำหนึ่งคำสั่ง (หรือกลุ่มคำสั่ง) เมื่อตรวจสอบเงื่อนไขแล้วเป็นเท็จ สามารถเขียนในรูป flowchart ได้ดังรูป



ภาพที่ 2 แสดง flowchart

รูปแบบของคำสั่ง if สองทางเลือก

รูปแบบการทำ 1 คำสั่ง : if (condition)

```
        Statement T;
else
        Statement F;
```

รูปแบบการทำ กลุ่มคำสั่ง : if (condition)

```
{
        Statement T;
        Statement T2;
}
else
{
        Statement F;
        Statement F2;
}
```

ตัวอย่างที่ 2.2 จงเขียนโปรแกรมรับข้อมูลเลขจำนวนเต็มทางแป้ นพิมพ์ แล้วทำการเปรียบเทียบกับเลขจำนวนเต็ม 100 ถ้าตัวเลขที่รับเข้ามานั้นมากกว่าให้แสดงข้อความ That number is greater than 100 ทางจอภาพ ถ้าตัวเลขที่รับเข้ามานั้นน้อยกว่าให้แสดงข้อความ That number is not greater than 100

```
#include <iostream>
using namespace std;
int main()
{
	int x;
	cout << "\nEnter a number: ";
	cin >> x;
	if( x > 100 )
		cout << "That number is greater than 100\n";
	else
		cout << "That number is not greater than 100\n";
	return 0;
}
```

ผลลัพธ์

```
Enter a number: 200
That number is greater than 100
```

2.1.3 หลายทางเลือก (Multiple-Alternative)

ในกรณีที่ตรวจสอบเงื่อนไขแล้ว ทำให้เกิดทางเลือกของการทำคำสั่งมากกว่า 2 ทางเลือกสามารถใช้คำสั่งแบบทางเลือก if แบบหลายทางเลือกได้ โดยเปลี่ยนการโปรแกรมให้อยู่ในรูปแบบหลายทางเลือกดังนี้

รูปแบบของคำสั่ง if หลายทางเลือก

```
if (condition)
{
    Statement;
}
    else if (condition2)
{
    Statement2;
}
else
```

```
{
    Statemen3t;
}
```

ตัวอย่างที่ 2.3 จงเขียนโปรแกรมรับข้อมูลเลขจำนวนเต็มทางแป้นพิมพ์ แล้วทำการเปรียบเทียบกับเลขจำนวนเต็ม 100 ถ้าตัวเลขที่รับเข้ามานั้นมากกว่าให้แสดงข้อความ That number is greater than 100 ทางจอภาพ ถ้าตัวเลขที่รับเข้ามานั้นน้อยกว่าให้แสดงข้อความ That number is not greater than 100 ถ้าตัวเลขที่รับเข้ามานั้นเท่ากัน ให้แสดงข้อความ That number is equal 100

```
#include <iostream>
using namespace std;
int main()
{
        int x;
        cout << "\nEnter a number: ";
        cin >> x;
        if( x > 100 )
                cout << "That number is greater than 100\n";
        else if ( x < 100 )
                cout << "That number is not greater than 100\n";
        else
                cout << "That number is equal 100\n";
        return 0;
}
```

ผลลัพธ์

```
Enter a number: 100
That number is equal 100
```

การใช้ฟังก์ชัน switch

คำสั่งแบบหลายทางเลือกโดยใช้คำสั่ง switch จะถูกใช้บ่อยในกรณีที่การเลือกในหลาย ๆ ทางเลือก เป็นลักษณะเชิงบังคับ ถ้าตรงกับกรณี(case) ใด ก็จะทำตามประโยคคำสั่งกรณีนั้นโดยค่าที่ใช้ตรวจสอบ (Controlling Expression) จะต้องเป็นข้อมูลชนิดจำนวนเต็ม (int) หรือข้อมูลชนิดตัวอักษร (char) โดยอาจจะอยู่ในรูปของตัวแปร, นิพจน์ หรือฟังก์ชันก็ได้ โดยข้อมูลชนิดตัวอักษร(char) ค่าคงที่ของกรณี (case) จะต้องใส่เครื่องหมาย ' '

รูปแบบของคำส7ง switch

```
switch (controlling expression)
{
case lable_set1 :
statements1
break;
case lable_set2 :
statements2
break;
:
:
case lable_setn :
statementsn
break;
default :
statementsd
}
```

ตัวอย่างที่ 2.4 โปรแกรมแบบหลายทางเลือก โดยการตรวจสอบข้อมูลว่าเป็น 1 2 หรือ 3 โดยรับข้อมูลจากแป้นพิมพ์ แล้วแสดงผลที่ตามเงื่อนไขที่รับข้อมูลเข้ามา

```
#include <iostream>
using namespace std;
int main( )
{
    int x;
    cout<< "Enter x: ";
    cin>>x;
    switch (x)
    {
        case 1: cout<< "Entering case 1\n"; break;
        case 2: cout<< "Entering case 2\n"; break;
        case 3: cout<< "Entering case 3\n"; break;
        default : cout << "Enter number 1-3 only";
```

```
        }
return 0;
}
```

ผลลัพธ์

Enter x: 100

Enter number 1-3 only

2.2 คำสั่งแบบการทำซ้ำ (repetition Statement)

หมายถึงคำสั่งการเขียนโปรแกรมให้ทำงานซ้ำ ๆ และมีเงื่อนไขให้เลิกการทำงานซ้ำนั้นแบ่งออกได้

- คำสั่งแบบการทำซ้ำ for
- คำสั่งแบบการทำซ้ำ while
- คำสั่งแบบการทำซ้ำ do/while

2.2.1 คำสั่งแบบการทำซ้ำ for

คำสั่งแบบการทำซ้ำ for เป็นคำสั่งที่ใช้สำหรับทำซ้ำคำสั่ง โดยการทำงานจะขึ้นอยู่กับองค์ประกอบที่สำคัญ 3 ส่วน

1. ค่าเริ่มต้นของตัวแปรควบคุมการทำซ้ำ (initialization expression)
2. ส่วนตรวจสอบเงื่อนไข (test expression)
3. ส่วนเปลี่ยนแปลงค่าของตัวแปรควบคุมการทำซ้ำ (update expression)

รูปแบบของคำสั่ง for

รูปแบบการทำ 1 คำสั่ง

```
for (initialization expression; test expression ; update expression)
        statements;
```

รูปแบบการทำกลุ่ม คำสั่ง

```
for (initialization expression; test expression ; update expression)
{
        statements 1;
        statements 2;
        :
        :
        statements n;
}
```

ตัวอย่างที่ 2.5 จงเขียนโปรแกรมเพื่อคำนวณหาผลลัพธ์ของเลขจำนวนเต็ม 0 ถึง 9 ยกกำลังสองและให้แสดงผลลัพธ์ทางจอภาพด้วย

```
#include <iostream>
using namespace std;
int main()
{
	int j;
	for(j=0; j<10; j++)
		cout << j * j << " ";
		cout << endl;
	return 0;
}
```

ผลลัพธ์

0 1 4 9 16 25 36 49 64 81

2.2.2 คำสั่งแบบการทำซ้ำ while

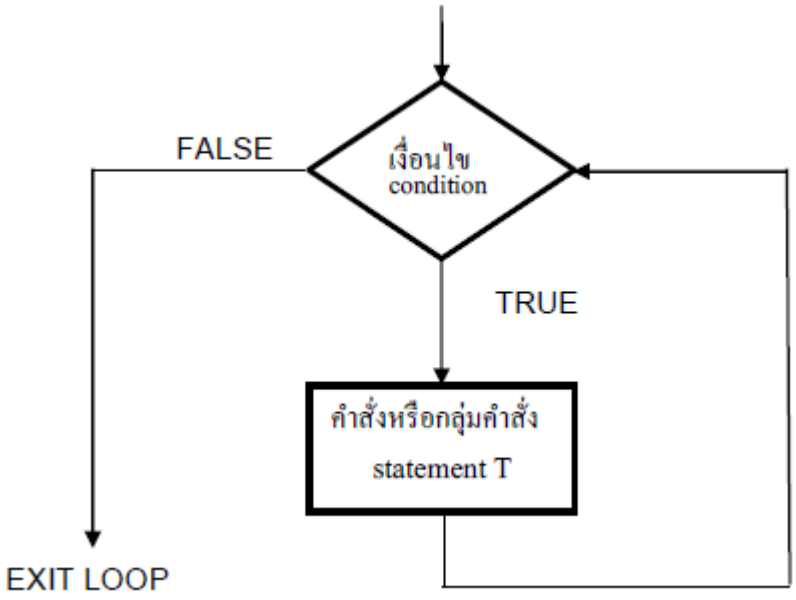
คำสั่งแบบการทำซ้ำ while เป็นคำสั่งที่ใช้สำหรับทำซ้ำคำสั่ง โดยการทำงานของการทำซ้ำ มีหลักการทำงานคือ

1. กำหนดค่าเริ่มต้นของตัวแปรควบคุมการทำซ้ำก่อน (initialization expression)
2. ตรวจสอบเงื่อนไขของคำสั่ง while (test expression)
3. กรณีเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่ง while พร้อมทั้งเปลี่ยนแปลงค่าเริ่มต้นของตัวแปรควบคุมการทำซ้ำที่อยู่ใน while กรณีเงื่อนไขเป็นเท็จก็ไม่ต้องทำงานในคำสั่งใน while

รูปแบบของคำสั่ง while

```
initialization expression;
	while (test expression) {
				statements;
				update expression;
				}
```

สามารถเขียนในรูป flowchart ได้ดังรูป



ตัวอย่างที่ 2.6 จงเขียนโปรแกรมแสดงเลขจำนวนเต็มจากมากไปน้อย โดยรับข้อมูลเลขจำนวนเต็มเริ่มต้นจากแป้นพิมพ์ และแสดงผลลัพธ์ทางจอภาพด้วย

```
#include <iostream>
using namespace std;
int main()
{
int n;
cout << "Enter the starting number > ";
cin >> n;
while (n>0) {
cout << n << ", ";
--n;
}
cout << "FIRE!";
return 0;
}
```

ผลลัพธ์

```
Enter the starting number > 5
5, 4, 3, 2, 1, FIRE!
```

2.2.3 คำสั่งแบบการทำซ้ำ do/while

คำสั่งแบบการทำซ้ำ while และ for จะต้องมีการตรวจสอบค่าของเงื่อนไขก่อนว่าเป็นจริงหรือเท็จ ก่อนที่จะทำคำสั่งภายใน while หรือ for แต่ในการทำงานบางลักษณะจะต้องทำคำสั่งภายในก่อน 1 ครั้ง แล้วจึงตรวจสอบค่าของเงื่อนไขว่าเป็นจริงหรือเท็จ ซึ่งการทำงานในลักษณะที่กล่าวมานี้สามารถใช้คำสั่ง do-while ได้ มีรูปแบบของคำสั่ง ดังนี้

รูปแบบของคำสั่ง do..while

```
initialization expression
do {
        statements;
        update expression; }
        while (test expression) ;
```

ตัวอย่างที่ 2.7 จงเขียนโปรแกรมแสดงเลขจำนวนเต็มจากมากไปน้อย โดยรับข้อมูลเลขจำนวนเต็มเริ่มต้นจากแป้นพิมพ์ และแสดงผลลัพธ์ทางจอภาพด้วย

```
#include <iostream>
using namespace std;
int main()
{
        int n;
        cout << "Enter the starting number > ";
        cin >> n;
        do {
                cout << n << ", ";
                --n;
        }
        while (n>0);
                cout << "FIRE!";
                return 0;
}
```

ผลลัพธ์

```
Enter the starting number > 5
5, 4, 3, 2, 1, FIRE!
```

2.3 การใช้คำสั่ง break , continue และ exit(0)

คำสั่ง break เป็นคำสั่งที่ใช้ในการหยุดการทำงานตามเงื่อนไขของคำสั่งแบบการทำซ้ำ

ตัวอย่างที่ 2.8 จงเขียนโปรแกรมแสดงเลขจำนวนเต็ม 1 ถึง 10 จากค่ามากไปค่าน้อย โดยเมื่อถึงเลข 3 ให้หยุดการแสดงเลขถัดไป และแสดงผลลัพธ์ที่ได้ทางจอภาพ

```
#include <iostream>
using namespace std;
int main ()
{
	int n;
	for (n=10; n>0; n--) {
	cout << n << ", ";
	if (n==3)
	{
		cout << "countdown aborted!";
		break;
	}
	}
return 0;
}
```

ผลลัพธ์

10, 9, 8, 7, 6, 5, 4, 3, countdown aborted!

คำสั่ง continue เป็นคำสั่งที่ใช้ในการหยุดการทำงาน ตามเงื่อนไขในรอบปัจจุบันแล้วให้ไปทำงานตามคำสั่งในรอบถัดไป ของคำสั่งแบบการทำซ้ำ

ตัวอย่างที่ 2.9 จงเขียนโปรแกรมแสดงเลขจำนวนเต็ม 1 ถึง 10 จากค่ามากไปค่าน้อย โดยข้ามเลขจำนวนเต็ม 5 และแสดงผลลัพธ์ที่ได้ทางจอภาพ

```
#include <iostream>
using namespace std;
int main ()
{
	for (int n=10; n>0; n--) {
```

```
            if (n==5) continue;
            cout << n << ", ";
            }
        cout << "FIRE!";
return 0;
}
```

ผลลัพธ์

10, 9, 8, 7, 6, 4, 3, 2, 1, FIRE!

คำสั่ง exit(0) เป็นคำสั่งที่ใช้ในการหยุดการทำงานของโปรแกรมตามเงื่อนไขที่กำหนด

ตัวอย่างที่ 2.10 จงเขียนโปรแกรมแสดงเลขจำนวนเต็ม 1 ถึง 10 จากค่ามากไปค่าน้อย โดยเมื่อถึงเลข 5 ให้หยุดการทำงานของโปรแกรม และแสดงผลลัพธ์ที่ได้ก่อนหยุดการทำงานของโปรแกรมทางจอภาพ

```
#include <iostream>
using namespace std;
int main ()
{
        for (int n=10; n>0; n--) {
        if (n==5) exit(0);
        cout << n << ", ";
        }
cout << "FIRE!";
return 0;
}
```

ผลลัพธ์

10, 9, 8, 7, 6,

สรุป

คำสั่งควบคุมการทำงานของโปรแกรมคำสั่งทางเลือก หมายถึงคำสั่งในการเขียนโปรแกรมให้มีการตัดสินใจเลือกที่จะทำหรือไม่ทำตามคำสั่งเงื่อนไขท7กำหนดไว้ ประกอบด้วยคำสั่งต่อไปนี้ คำสั่ง if คำสั่ง if..else คำสั่ง if..else if คำสั่ง switch

คำสั่ง ควบคุมการทำงานของโปรแกรมทำซ้ำ หมายถึงคำสั่งในการเขียนโปรแกรมให้มีการกระทำซ้ำ ๆ และมีการกำหนดเงื่อนไขให้เลิกการกระทำซ้ำ ประกอบด้วยคาสั่งต่อไปนี้ คาสั่ง for คำสั่ง while คำสั่ง do..while

คำสั่ง break คำสั่ง continue และคำสั่ง exit(0) เป็นคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรมทำซ้ำโดยกำหนดเงื่อนไขในการใช้คำสั่ง

คำสั่ง break ใช้ในการหยุดการทำคำสั่งภายในรอบของคำสั่งแบบการทำซ้ำ

คำสั่ง continue เป็นคำสั่ง ที่ใช้ในการหยุดการทำคำสั่งภายในรอบปัจจุบันแล้วให้ไปทำงานในรอบถัดไป

คำสั่ง exit(0) เป็นคำสั่ง ที่ใช้ในการหยุดการทำงานของโปรแกรม