บทที่ 9 สายอักขระในภาษาซีพลัสพลัสและการจัดการไฟล์ (C++ String and File Processing)

วัตถุประสงค์

1) เข้าใจการทำงานกับสายอักขระในภาษาซีพลัสพลัส

2) เขียนโปรแกรมจัดการข้อมูลสายอักขระได้

3) เขียนโปรแกรมเพื่ออ่านข้อมูลจากแฟ้มข้อมูล และบันทึกข้อมูลในแฟ้มข้อมูลได้

ในบทนี้เราจะเรียนสองเรื่อง เรื่องแรกคือการทำงานกับสายอักขระในภาษาซีพลัสพลัส ซึ่งเราเรียนบางส่วนไปแล้วในบทพื้นฐานการเขียนภาษาซีพลัสพลัส แต่เราจะเพิ่มเติมเรื่องการใช้ฟังก์ชันและคุณสมบัติของอาเรย์ในการจัดการสายอักขระ ส่วนเรื่องที่สองคือการอ่านข้อมูลจากแฟ้มข้อมูล (file) และการเขียนข้อมูลลงในแฟ้มข้อมูล ซึ่งคำสั่งในการทำงานทั้งสองอย่างนี้จะคล้ายกับคำสั่งการอ่านข้อมูลผ่านทางแป้นพิมพ์ (cin) และการแสดงข้อมูลออกทางหน้าต่างคอนโซล (cout) มาก แต่ต่างกันที่เราต้องใช้ตัวแปรแบบ filestream ในการอ่านและเขียนข้อมูล

9.1 สายอักขระในภาษาซีพลัสพลัส (C++ String)

สายอักขระ หรือ สตริง (string) คือข้อมูลที่ประกอบด้วยอักขระจำนวน 0 ตัวหรืออย่างน้อย 1 ตัว เราเรียกสายอักขระที่มีอักขระ 0 ตัวว่าสตริงว่าง (empty string) ฟังก์ชันที่ใช้ทำงานกับสตริงนั้นได้ถูกนิยามในไลบรารี <string> เนื่องจากสตริงเป็นอักขระที่มาเรียงต่อกันหลายๆ ตัว เราจึงสามารถทำงานกับสตริงเสมือนว่ามันเป็นแถวลำดับของอักขระ (array of characters) ได้ อักขระแต่ละตัวจึงมีเลขระบุตำแหน่งแบบอาเรย์ นั่นคือตำแหน่งแรกคือตำแหน่งที่ 0 แต่ตำแหน่งสุดท้ายคือตำแหน่งที่ n-1 เมื่อ n เป็นความยาวของสตริงหรือจำนวนอักขระที่อยู่ในสตริงนั่นเอง การประกาศสตริงนั้นทำได้หลายแบบดังที่ได้อธิบายไว้ในเรื่องตัวแปรและชนิดข้อมูลในบทที่ 4 แล้ว เราลองมาดูตัวอย่างการประกาศสตริงที่จะทำให้เห็นภาพการเป็นอาเรย์ของอักขระเมื่อเราประกาศตัวแปรต่อไปนี้

```
string str3 = "New York";
string str6(str3, 4, 2);
```

ในการให้ค่าตัวแปร str6 นั้น โปรแกรมจะไปดูที่ str3 จากนั้นจะนำอักขระใน str3 จำนวน 2 ตัว โดยเริ่มจากตำแหน่งที่ 4 มาสร้างเป็น str6 ในตัวอย่างนี้ อักขระตำแหน่งที่ 0 คือ 'N' ดังนั้นอักขระตำแหน่งที่ 4 คือ 'Y' (นับรวมเว้นวรรคเป็น 1 อักขระ) เมื่อโปรแกรมเอาอักขระมา 2 ตัว เราก็จะได้ "Yo" มาใส่ในตัวแปร str6

ตัวแปรชนิดสตริงในภาษาซีพลัสพลัสนั้นมีเมธอด (method) ให้เรียกใช้ เมธอดจะทำงานคล้ายฟังก์ชันแต่จะถูกผูกติดกับวัตถุ (ในที่นี้คือตัวแปรที่เป็นชนิดสตริง) และทำงานได้กับเฉพาะชนิดของวัตถุที่กำหนดไว้เท่านั้น โดยวิธีใช้เมธอดนั้น เราจะเขียนตัวแปรชนิดสตริง ตามด้วยจุด และเมธอดและวงเล็บ (ซึ่งจะมีพารามิเตอร์หรือไม่มีก็ได้) ตามลำดับ เช่น str.method1() ในบทเรียนนี้ เราจะยกตัวอย่างเมธอดของสตริง ดังตารางที่ 15

ตารางที่ 15 เมธอด (method) และฟังก์ชัน (function) ที่ใช้กับสตริง (string)

คำสั่งที่ใช้	**ชนิดข้อมูลผลลัพธ์**	**รายละเอียด**
str.length()	unsigned int	หาความยาวของสตริง (จำนวนอักขระ – รวมเว้นวรรค เครื่องหมาย และตัวเลขต่างๆ)
str.append(string1)	string	เอา string1 ต่อเข้ากับ str * สตริง str จะเปลี่ยนค่าไปหลังจากรันคำสั่งนี้
getline(cin, str)	ชนิดเดียวกับ cin (เราสามารถเปลี่ยน cin เป็น istream ชนิดอื่นได้ เช่น ifstream ที่จะได้เรียนในส่วนที่สองของบทนี้)	อ่านข้อความทั้งบรรทัดจากแป้นพิมพ์ (รวมทั้งเว้นวรรคด้วย) เข้ามาเก็บไว้ที่ตัวแปร str * ถ้าเราใช้คำสั่ง cin >> str โปรแกรมจะอ่านมาแค่ข้อความวรรคแรกเท่านั้น เช่น ถ้าเราใส่ This is a book. ในคอนโซลแล้วเราใช้คำสั่ง cin >> str โปรแกรมจะอ่านแค่ This มาเก็บไว้ แต่ถ้าเราใช้ getline(cin, str) โปรแกรมจะอ่าน This is a book. มาเก็บไว้ใน str ** จริงๆ คำสั่งนี้เป็นฟังก์ชันที่ใช้กับสตริง ไม่ใช่เมธอด
str.substr(position, len)	string	ทำสำเนาของสตริง str ไปเก็บไว้ที่สตริงอื่นโดยเริ่มจากตำแหน่งที่ position และทำสำเนาอักขระจำนวน len ตัว
str.find(string1, position)	unsigned int	หาสตริง string1 ที่อยู่ใน str โดยเริ่มจากตำแหน่งที่ position ถ้าหาเจอให้คืนค่าตำแหน่งของอักขระแรกของ string1 แต่ถ้าหาไม่เจอให้คืนค่า npos (ในที่นี้คือค่าสูงสุดของ unsigned size_t คือ 4,294,967,295)

str.find(string1)	unsigned int	เหมือนกับคำสั่งข้างบน แต่ให้เริ่มหาที่ตำแหน่งที่ 0
str.erase(position, len)	string	ลบส่วนของสตริง str โดยเริ่มจากตำแหน่งที่ position และลบอักขระไปจำนวน len ตัว * สตริง str จะเปลี่ยนค่าไปหลังจากรันคำสั่งนี้
str.replace(position, len, string1)	string	เอา string1 ไปแทนที่ส่วนของสตริง str โดยแทนที่เริ่มจากตำแหน่ง position และแทนที่อักขระเป็นจำนวน len ตัว * สตริง str จะเปลี่ยนค่าไปหลังจากรันคำสั่งนี้

ตัวอย่างการใช้เมธอดและฟังก์ชันที่ใช้กับสตริง

เมธอด length() ใช้หาความยาวของสตริงซึ่งคือจำนวนอักขระในสตริงนั้น จากตัวอย่างโค้ดข้างล่างนี้ สตริง s มีอักขระ 7 ตัวต่อกัน จึงมีความยาวเท่ากับ 7 ส่วนสตริง ms นั้นมีความยาว 15 โดยเวลานับอักขระให้นับ “อักขระ” ไม่ใช่ “อักษร” ดังนั้นเราจึงต้องนับช่องว่าง เครื่องหมาย ตัวเลข และอักขระที่ไม่แสดงออกด้วยเช่น ‘\t’ และ ‘\n’

โค้ด:

```
string s = "ABCDEFG";
cout << s.length() << endl;
string ms = "This is a book.";
cout << ms.length() << endl;
```

ผลการทำงาน:

```
7
15
```

เมธอด append จะต่อสตริงที่เป็นพารามิเตอร์ของเมธอดเข้ากับสตริงหลัก โดยที่หลังจากรันคำสั่งแล้ว สตริงหลักจะถูกเปลี่ยนค่าไปเป็นค่าที่ถูกต่อแล้ว ดังตัวอย่างในโค้ดข้างล่างนี้ สตริง s ถูกต่อด้วยสตริง “XYZ” และ

กลายเป็นมีค่า “ABCDEFGXYZ” แต่ถ้าเราต่อสตริงด้วยตัวดำเนินการ + สตริงที่ถูกต่อ (สตริง s) จะไม่มีการเปลี่ยนค่า แต่สตริงผลลัพธ์ (“ABCDEFGXYZHIJK”) จะถูกเก็บไว้ที่สตริงอื่น (สตริง s2) แทน

โค้ด:

```
string s = "ABCDEFG";
string s1 = s.append("XYZ");
cout << "s1 = " << s1 << endl;
cout << "s = " << s << endl;
string s2 = s + "HIJK";
cout << "s2 = " << s2 << endl;
cout << "s = " << s << endl;
```

ผลการทำงาน:

```
s1 = ABCDEFGXYZ
s = ABCDEFGXYZ
s2 = ABCDEFGXYZHIJK
s = ABCDEFGXYZ
```

ฟังก์ชัน getline() จะทำการอ่านค่าสตริงที่ถูกป้อนผ่านทางแป้นพิมพ์ทั้งบรรทัดโดยรวมทั้งเว้นวรรคด้วย ซึ่งจะแตกต่างจากคำสั่ง cin เฉยๆ ในกรณีที่เราใช้ cin >> str โปรแกรมจะอ่านสตริงจนถึงเว้นวรรคเท่านั้น นั่นคือจะอ่านเข้ามาคำเดียว แต่คำสั่ง getline(cin, str) จะอ่านอินพุตเข้ามาทั้งบรรทัด ดังตัวอย่างข้างล่างนี้

โค้ด:

```
string s1,s2,s3;
cout << "Enter text: ";
cin >> s1 >> s2;
getline(cin, s3);
cout << s1 << endl << s2 << endl << s3 << endl;
```

ผลการทำงานเมื่อผู้ใช้ป้อน

```
Hello World
This is me.
```

ผ่านทางแป้นพิมพ์:

```
Hello
World
This is me.
```

นั่นคือตัวแปร s1 จะเก็บค่าข้อความ “Hello” ส่วนตัวแปร s2 จะเก็บค่าข้อความ “World” และตัวแปร s3 ซึ่งให้คำสั่ง getline() รับค่าเข้ามาจะเก็บค่า “This is me.”

เมธอด substr() จะทำสำเนา (copy) ส่วนของสตริงหลักไปใส่ในสตริงอื่น โดยจะเริ่มทำสำเนาจากตำแหน่งที่กำหนดและทำสำเนาอักขระตามจำนวนที่กำหนด เช่นในโค้ดข้างล่างนี้ คำสั่ง s.substr(5, 3) จะทำสำเนาโดยเริ่มจากตำแหน่งที่ 5 ของสตริง s (ตัว ‘F’) และทำสำเนาอักขระเป็นจำนวน 3 ตัว ก็จะได้สตริงที่มีค่าเป็น “EFG” ออกมาเก็บไว้ในสตริง s1 โดยที่สตริง s ไม่มีการเปลี่ยนแปลงค่าใดๆ

โค้ด:

```
string s = "ABCDEFGHIJKLM";
string s1 = s.substr(5, 3);
cout << s1 << endl;
```

ผลการทำงาน:

```
FGH
```

เมธอด find() หาสตริงที่กำหนดเป็นพารามิเตอร์ในสตริงหลัก แล้วคืนค่า (return) ตำแหน่งของสตริงตัวแรกที่หาเจอ โดยตำแหน่งดังกล่าวจะเป็นตำแหน่งของอักขระตัวแรกในสตริงที่หาเจอนั้น แต่ถ้าหาไม่เจอให้คืนค่า string::npos คือค่า 4294967295 ซึ่งเป็นค่าที่มากที่สุดของข้อมูลชนิด size_t ในการหาสตริงในสตริงหลักนั้น เราจะกำหนดตำแหน่งแรกของอักขระในสตริงหลักที่จะให้เมธอดหาก็ได้หรือไม่กำหนดก็ได้ หากไม่ได้กำหนดเมธอด find() จะเริ่มหาสตริงที่ตำแหน่งแรก (ตำแหน่งที่ 0) ของสตริงหลัก เช่น ถ้าเราเขียนคำสั่ง s7.find("si")

เราหมายความว่าให้หาสตริง “si” ที่อยู่ใน s7 โดยเริ่มจากตำแหน่งแรก ส่วนคำสั่ง s7.find("si",5) หมายความว่าให้หาสตริง “si” ที่อยู่ใน s7 โดยเริ่มจากอักขระตำแหน่งที่ 5 (ตัวที่ 6) ของสตริง s7

โค้ด:

```
string s7 = "Mississippi River Basin";
cout << s7.find("si",0) << endl;
cout << s7.find("si") << endl;
cout << s7.find("si",5) << endl;
cout << s7.find("so",5) << endl;
```

ผลการทำงาน:

```
3
3
6
4294967295
```

เมธอด erase() จะทำการลบส่วนของสตริงหลัก โดยจะเริ่มลบที่ตำแหน่งที่ระบุในพารามิเตอร์ตัวแรก และลบอักขระตามจำนวนที่ระบุในพารามิเตอร์ตัวที่สอง หลักจากรันคำสั่งนี้แล้ว สตริงหลักจะถูกเปลี่ยนไป (จะถูกลบบางส่วนออกไป) เพราะฉะนั้นจะต้องระวังเวลาใช้ เช่นในตัวอย่างข้างล่างนี้ คำสั่ง s6.erase(4,2) จะทำการลบส่วนของสตริงโดยเริ่มจากตำแหน่งที่ 4 ซึ่งคือตัว E และทำการลบไป 2 ตัว ก็คือ “EF” เมื่อเราแสดงผลของ s6 ออกทางหน้าจอ เราจะได้ s6 เป็นสตริง “ABCDGHIJK” (ตัว EF หายไป)

เมธอด replace() จะทำงานคล้ายกับคำสั่ง erase() แต่มันจะแทนที่สตริงที่ถูกลบไปด้วยสตริงที่กำหนดให้ในพารามิเตอร์ตัวที่สาม เช่นคำสั่ง s6.replace(5, 2, "xyz") จะทำการลบสตริงโดยเริ่มจากตำแหน่งที่ 5 ของ s6 (ที่ถูกเปลี่ยนเป็น “ABCDGHIJK” จากคำสั่งก่อนหน้านี้แล้ว) และลบไป 2 ตัว ซึ่งคือ “HI” จากนั้นมันจะนำสตริง “xyz” มาใส่แทนที่ และได้ผลลัพธ์เป็น “ABCDGxyzJK”

โค้ด:

```
string s6 = "ABCDEFGHIJK";
s6.erase(4, 2);
cout << s6 << endl;
s6.replace(5, 2, "xyz");
cout << s6 << endl;
```

```
```

ผลการทำงาน:

```
ABCDGHIJK
ABCDGxyzJK
```

ตัวดำเนินการที่ใช้กับสตริง

เราสามารถใช้ตัวดำเนินการ 2 แบบกับสตริง คือการเปรียบเทียบสตริงโดยใช้เครื่องหมาย ==, !=, >, >=, < หรือ <= และการต่อสตริงโดยใช้เครื่องหมาย + เมื่อเราเปรียบเทียบสตริง 2 สตริง โปรแกรมจะทำการเปรียบเทียบอักขระในสตริงคู่นั้นไปทีละตำแหน่งโดยใช้ค่า ASCII ของอักขระนั้น เมื่อพบตำแหน่งที่อักขระไม่เหมือนกัน โปรแกรมก็จะดูว่าค่า ASCII ของอักขระใดมากกว่าก็จะให้ผลลัพธ์ว่าสตริงตัวนั้นมีค่ามากกว่า เช่น เมื่อเปรียบเทียบ “Hello” และ “HelLo” มันจะพบว่าสตริงคู่นี้แตกต่างกันที่ตำแหน่งที่ 3 เป็นตำแหน่งแรก แล้วค่า ASCII ของ L มีค่ามากกว่า l จึงกำหนดให้ “HelLo” มีค่ามากกว่า “Hello”

โค้ด:

```
string s1 = "Somchai";
string s2 = "Somjai";
if (s1 < s2)
   cout << s1 << " comes first." << endl;
```

ผลการทำงาน:

```
Somchai comes first.
```

โค้ด:

```
string s3;
string s4 = "ILoveYou.";
```

```
cin >> s3;
while (s4 == s3)
{
   cout << "I love you too!" << endl;
   cin >> s3;
}
```

ผลการทำงาน เมื่อผู้ใช้ป้อน

ILoveYou.

I Love You.

ผ่านทางแป้นพิมพ์ตามลำดับ:

```
I love you too!
```

ข้อความ “I love you too!” เป็นข้อความที่เกิดจากอินพุตตัวแรก (s3 เหมือนกับ s4) แต่เนื่องจากอินพุตตัวที่สองนั้นมีการเว้นวรรค การใช้ cin อ่านเข้ามาจะอ่านได้แค่ “I” เท่านั้น ทำให้ s3 ไม่เท่ากับ s4 ในการวนซ้ำรอบที่สอง ดังนั้นโปรแกรมจึงหยุดการทำงาน

ส่วนการต่อสตริงด้วยเครื่องหมาย + นั้นจะให้ผลการทำงานเหมือนกับเมธอด append() นั่นคือโปรแกรมจะนำสตริงที่อยู่ด้านหลังไปต่อสตริงที่อยู่ด้านหน้า เช่น string s1 = “Yam” + s2; โดยที่ s2 มีค่าเป็น “Strawberry” จะทำให้ s1 มีค่าเป็น “YamStrawberry” ในตัวอย่างข้างล่างนี้ การให้ค่า s5 นั้นตรงไปตรงมา แต่สำหรับ s2 นั้น เราต้องต่อสตริงของ s5 เข้าข้างหลัง s2 ไม่ใช่ข้างหน้า เพราะคำสั่ง s2 += s5 นั้นหมายความว่า s2 = s2 + s5 (s2 มาก่อนเสมอ) เราจึงได้สตริง “XYZ” มานำหน้าสตริง “ABCDEFGHIJK” และกลายเป็น “XYZABCDEFGHIJK”

โค้ด:

```
string s = "ABCDEFG";
string s5 = s + "HIJK";
cout << s5 << endl;

string s2 = "XYZ";
s2 += s5;
cout << s2 << endl;
```

ผลการทำงาน:

```
ABCDEFGHIJK
XYZABCDEFGHIJK
```

อาเรย์ของอักขระ (array of characters)

เนื่องจากสตริงประกอบด้วยอักขระหลายๆ ตัวมาเรียงต่อกัน เราสามารถทำงานกับสตริงโดยคิดว่ามันเป็นอาเรย์ของอักขระก็ได้ นั่นคือเราสามารถเข้าถึงอักขระในตำแหน่งต่างๆ ของสตริงโดยใช้รูปแบบของอาเรย์ เช่นในสตริง s1 = “Kornchawal” นั้น s1[4] จะหมายถึงอักขระในตำแหน่งที่ 4 (ตัวที่ 5) นั่นคือตัว c ดังนั้นถ้าเราแสดงผล cout << s1[4] เราจะได้ c ออกมาทางหน้าจอ และถ้าเรารันคำสั่ง s1[4] = “C” เราก็จะได้ค่า s1 ที่เปลี่ยนแปลงไปเป็น “KornChawal” นั่นคือตำแหน่งที่ 4 จะถูกแทนที่ด้วยอักขระ C ใหญ่

โค้ด:

```
string s = "ABCDEFG";
char c = s[4];
cout << c << endl;

s[2] = "*";
cout << s << endl;
```

ผลการทำงาน:

```
E
AB*DEFG
```

ตัวอย่างโจทย์สตริง: จงหาผลการทำงานของโปรแกรมต่อไปนี้

โค้ด:

```
int main(){
  string S1 = "Sawasdee Ja";
```

```
  string S2 = "Somsri Konkeng";
  string S3 = "He is a student of KhonKaen University";
  string S4;
  cout << S3.length() << endl;
  S4 = S2 + S3.substr(2, 13);
  cout << S4 << endl;
  cout << S3.find("en", 0) << endl;
  S4 = S1.replace(9, 2, S2.erase(7,7));
  cout << S4 << endl;
  cout << (S2 < "Somsak Kondee") << endl;
  return 0;
}
```

ผลการทำงาน:

```
38
Somsri Konkeng is a student
12
Sawasdee Somsri
0
```

โค้ดในตัวอย่างข้างบนนี้นำมาจากข้อสอบปลายภาค การแสดงผลบรรทัดแรกคือ S3.length() ซึ่งเป็นความยาวของสตริง S3 นั่นคือให้เรานับจำนวนอักขระ (รวมตัวอักษร ช่องว่าง และสัญลักษณ์ต่างๆ) เราก็จะได้ 38 ตัว คำสั่งที่สองคือการให้ค่า S4 โดยการต่อสตริง โดยต่อ S2 กับส่วนของสตริง S3 โดยเริ่มที่ตำแหน่งที่ 2 และทำสำเนามา 13 ตัว คำสั่ง S3.substr(2, 13) จะได้ผลลัพธ์เป็น “ is a student” (มีช่องว่างอยู่ข้างหน้า) เมื่อต่อกับ S2 ก็จะได้ “Somsri Konkeng is a student”

เอาต์พุตต่อมาเป็นผลลัพธ์ของคำสั่ง S3.find("en", 0) ซึ่งเป็นการหาสตริง “en” ในสตริงหลัก S3 โดยเริ่มจากตำแหน่งแรก “en” ตัวแรกที่เราหาเจออยู่ในคำว่า student นั่นคือที่ตำแหน่งที่ 12 ซึ่งเป็นตำแหน่งของอักขระ e ส่วนเอาต์พุตต่อมาจะซับซ้อนเล็กน้อย เพราะเป็นการเรียกฟังก์ชันซ้อนฟังก์ชัน สิ่งที่เราต้องทำคือทำฟังก์ชันในสุดก่อน ซึ่งคือคำสั่ง S2.erase(7,7) ในคำสั่งนี้เราจะลบสตริงออกจาก S2 โดยเริ่มจากตำแหน่งที่ 7 และลบไป 7 อักขระ นั่นคือลบ “Konkeng” ออก และได้ผลลัพธ์เป็น “Somsri ” (มีช่องว่างด้านหลังสุด) จากนั้นเราจึงทำคำสั่ง S1.replace(9, 2, “Somsri ”) ซึ่งจะลบอักขระ 2 ตัวในสตริง S1 โดยเริ่มจากตำแหน่งที่ 9 นั่นคือลบ “Ja” ออกไป แล้วแทนที่ “Ja” ด้วย “Somsri ” เราก็จะได้ S4 เป็น “Sawasdee Somsri ”

ส่วนคำสั่งสุดท้ายนั้นเป็นการแสดงผลการเปรียบเทียบว่า S2 (ที่ถูกเปลี่ยนค่าไปแล้วจากคำสั่ง erase ก่อนหน้านี้) ซึ่งคือ “Somsri ” มีค่าน้อยกว่าสตริง "Somsak Kondee" ใช่หรือไม่ เราทำการเปรียบเทียบไปทีละคู่อักขระและพบว่าอักขระที่แตกต่างกันตัวแรกคือ r ใน "Somsri " และ a ใน “Somsak Kondee” จากค่า ASCII เรารู้ว่าค่าของ r มีค่ามากกว่า a ทำให้ค่าของสตริง "Somsri " มีค่ามากกว่า “Somsak Kondee” ทำให้การเปรียบเทียบที่โจทย์กำหนดให้นั้นเป็นเท็จ โปรแกรมจึงแสดงค่า 0 ออกทางหน้าจอ

9.2 การจัดการไฟล์ (File Processing)

ที่ผ่านมาเรารับข้อมูลนำเข้าจากแป้นพิมพ์และส่งข้อมูลออกทางหน้าจอ เราสามารถอ่านข้อมูลนำเข้าจากไฟล์และส่งออกข้อมูลลงไปในไฟล์ (เขียนลงไฟล์) ได้เช่นเดียวกัน โดยลักษณะการใช้งานจะเหมือนการใช้คำสั่ง cin และ cout มาก ยกเว้นว่าถ้าเราจะอ่านข้อมูลนำเข้าจากไฟล์หรือส่งออกข้อมูลลงไปในไฟล์ เราจะต้องทำการเปิดไฟล์นั้นเพื่ออ่านหรือเพื่อเขียนก่อนตามรายละเอียดต่อไปนี้

การอ่านข้อมูลหรือการส่งออกข้อมูลจะถูกควบคุมด้วยวัตถุ (object) ที่เป็นสตรีม (stream) ในกรณีของคำสั่ง cin และ cout นั้นการอ่านข้อมูลและการส่งออกข้อมูลจะถูกควบคุมด้วย istream (input stream) และ ostream (output stream) ตามลำดับ (หมายความว่า cin จริงๆ แล้วเป็นวัตถุชนิด istream และ cout เป็นวัตถุชนิด ostream) จะสังเกตได้จากเราต้องอ้างอิงหรือ include ไลบรารีที่ชื่อว่า iostream ในทุกไฟล์ ส่วนการอ่านข้อมูลจากไฟล์นั้นจะถูกควบคุมด้วย ifstream (input file stream) และการส่งออกข้อมูลลงไฟล์นั้นจะถูกควบคุมด้วย ofstream (output file stream) ดังนั้นเวลาที่เราต้องการทำงานกับไฟล์ในลักษณะดังกล่าว เราจะต้องอ้างอิงไลบรารีที่ชื่อว่า fstream (file stream) ลงไปในไฟล์ด้วย

ก่อนอ่านหรือเขียนลงไฟล์ เราจะต้องสร้างวัตถุเพื่อทำงานกับไฟล์ก่อน (หรือจะเรียกว่าการเปิดไฟล์เพื่อทำงานก็ได้) วัตถุที่เราสร้างจะมีชนิดเป็น ifstream สำหรับการเปิดไฟล์เพื่ออ่านข้อมูล และจะมีชนิดเป็น ofstream จากนั้นเราต้องระบุชื่อวัตถุที่เราจะใช้ในการอ่านหรือเขียนลงไฟล์ (นั่นคือใช้แทนคำสั่ง cin หรือ cout) แล้วตามด้วยชื่อไฟล์ที่เราต้องการทำงานด้วย ตามตัวอย่างข้างล่างนี้

โค้ด:

```
#include <fstream> // เพิ่มเข้ามาจากหัวไฟล์ปกติ

int main(){
  ifstream myin(“input.txt”);
  ofstream myout(“output.txt”);
  …
}
```

ในตัวอย่างนี้ เราเปิดไฟล์ที่ชื่อว่า input.txt เพื่ออ่านข้อมูลเข้า โดยไฟล์ที่ชื่อว่า input.txt จะต้องอยู่ในโฟลเดอร์เดียวกับที่โปรแกรมเราอยู่ ไม่เช่นนั้นเราจะต้องใส่ path หรือตำแหน่งที่อยู่ของไฟล์ในคอมพิวเตอร์ของเราให้โปรแกรมของเราทราบด้วย ในการอ่านข้อมูลนั้นเราจะใช้วัตถุที่ชื่อว่า myin แทนคำสั่ง cin เช่น ถ้าเราต้องการอ่านข้อมูลเข้ามาเก็บในตัวแปร x จากแป้นพิมพ์เราจะใช้คำสั่งว่า cin >> x; แต่ถ้าเราต้องการอ่านข้อมูลจากไฟล์นี้เข้ามาเก็บในตัวแปร x เราจะใช้คำสั่งว่า myin >> x; แทน

ในโปรแกรมนี้ เรายังได้เปิดไฟล์เพื่อเขียนข้อมูลลงไปด้วย โดยเราเปิดไฟล์ที่ชื่อว่า output.txt ในการเขียนข้อมูลนั้นเราจะใช้วัตถุที่ชื่อว่า myout แทนคำสั่ง cout เช่น ถ้าเราต้องการเขียนข้อมูลของตัวแปร y ออกทางหน้าจอ เราจะใช้คำสั่งว่า cout << y; แต่ถ้าเราต้องการเขียนข้อมูลของตัวแปร y ลงในไฟล์นี้ เราจะใช้คำสั่งว่า myout << y; แทน

โค้ด:

```
#include <iostream>

int main()
{
  int value = 0;
  cin >> value;
  return 0;
}
```

```
#include <fstream>

int main()
{
  int value = 0;
  ifstream fin("number.dat");
  fin >> value;
  return 0;
}
```

โค้ดข้างบนนี้แสดงการเปรียบเทียบการส่งข้อมูลออกทางหน้าจอด้วยคำสั่ง cin และการอ่านข้อมูลจากไฟล์ที่ชื่อว่า number.dat ด้วยวัตถุที่ชื่อว่า fin เราจะเห็นได้ว่ารูปแบบของคำสั่งเหมือนกันมาก แต่ต่างกันตรงที่การอ่านข้อมูลจากไฟล์นั้นเราจะต้องเปิดไฟล์เพื่ออ่านก่อน แล้วอ่านข้อมูลเข้ามาด้วยวัตถุที่เราสร้างขึ้นมาตอนเปิดไฟล์นั่นเอง

โค้ด:

```
#include <iostream>

int main()
```

```
#include <fstream>

int main()
```

```
{
  string str = "my text";
  cout << str;
  return 0;
}
```

```
{
  string str = "my text";
  ofstream fout("myfile.dat");
  fout << str;
  return 0;
}
```

โค้ดข้างบนนี้แสดงการเปรียบเทียบการส่งข้อมูลออกทางหน้าจอด้วยคำสั่ง cout และการส่งข้อมูลออกไปยังไฟล์ที่ชื่อว่า myfile.dat ด้วยวัตถุที่ชื่อว่า fout เราจะเห็นได้ว่ารูปแบบของคำสั่งเหมือนกันมาก แต่ต่างกันตรงที่การส่งข้อมูลออกไปยังไฟล์จะต้องเปิดไฟล์เพื่อเขียนก่อน แล้วเขียนข้อมูลออกไปด้วยที่วัตถุที่เราสร้างขึ้นมาตอนเปิดไฟล์

สำหรับการเปิดไฟล์เพื่อเขียนหรืออ่านข้อมูล เราต้องสร้างวัตถุ 1 ชิ้นสำหรับการทำงานกับไฟล์ 1 อย่าง นั่นหมายความว่า ถ้าเราต้องการเปิดไฟล์เพื่ออ่าน 2 ไฟล์ เราจะต้องสร้างวัตถุขึ้นมา 2 ชิ้นในชื่อที่แตกต่างกัน โดยวัตถุแต่ละชิ้นจะใช้อ่านไฟล์ 1 ไฟล์เท่านั้น ในกรณีเดียวกัน ถ้าเราต้องการเปิดไฟล์เพื่อเขียน 2 ไฟล์ เราจะต้องสร้างวัตถุขึ้นมา 2 ชิ้นในชื่อที่แตกต่างกัน โดยวัตถุแต่ละชิ้นจะใช้เขียนลง 1 ไฟล์เท่านั้น

นอกจากนี้ การทำงานกับไฟล์ไม่มีข้อกำหนดว่าเมื่ออ่านจากไฟล์แล้วเราจะต้องเขียนลงไฟล์เท่านั้น เราสามารถเปิดไฟล์เพื่ออ่านข้อมูลแต่ส่งข้อมูลออกทางหน้าจอก็ได้ หรือรับข้อมูลเข้าทางหน้าจอแต่ส่งออกข้อมูลไปยังไฟล์ก็ได้ หรือเราสามารถทำการเขียนออกทางหน้าจอและเขียนลงไฟล์ในโปรแกรมเดียวกันก็ได้ เราก็แค่เขียนคำสั่ง 2 แบบต่อกัน เช่นถ้าเราต้องการเขียนค่า x ออกทางหน้าจอและเขียนค่า x ลงไฟล์ด้วย เราก็เขียนคำสั่งต่อกันดังนี้

```
cout << x;
fout << x;
```

ค่า x ก็จะไปปรากฏทั้งบนจอและในไฟล์ที่ fout ถูกเปิดขึ้นมาเพื่อเขียนนั้น

การเขียนคำสั่งให้โปรแกรมอ่านจนจบไฟล์

ในการเขียนโปรแกรมให้อ่านค่าเข้ามาเรื่อยๆ โดยใช้คำสั่ง cin เราสามารถวนลูปโดยใช้โค้ดต่อไปนี้ได้

โค้ด:

```
int main()
{
  int value;
  while(cin >> value)
    cout << "value = "<< value << endl;
  return 0;
}
```

เราสามารถเอาคำสั่ง cin ไปไว้เป็นเงื่อนไขในการลูป เพราะคำสั่ง cin นั้นจริงๆ แล้วมีการคืนค่า (return) เป็นชนิด boolean มันจะคืนค่าเป็นจริง (true) เมื่อมีการอ่านข้อมูลเข้ามาได้อย่างประสบความสำเร็จ และจะคืนค่าเป็นเท็จ (false) ถ้ามันไม่สามารถอ่านข้อมูลเข้ามาได้ (เช่น ถ้าเราต้องการอ่านข้อมูลเป็น int แต่เราใส่ข้อมูลเข้ามาเป็นข้อมูลชนิดที่ไม่ใช่ int) สำหรับการอ่านจากไฟล์ก็เช่นเดียวกัน เราสามารถเอาคำสั่งการอ่าน (เช่น fin >> x) เข้าไปเป็นเงื่อนไขในการวนลูปได้ สิ่งที่จะเกิดขึ้นก็คือโปรแกรมจะอ่านข้อมูลในไฟล์ไปเรื่อยๆ จนกว่าจะจบไฟล์ ซึ่งจะมีการคืนค่าเป็นเท็จถ้าจบไฟล์แล้ว เพราะมันอ่านต่อไม่ได้แล้ว ดังตัวอย่างต่อไปนี้

โค้ด:

```
int main()
{
  ifstream fin("number.dat");
  int value = 0;
  int sum = 0;
  while (fin >> value)
  {
     sum += value;
  }
  cout << "sum = " << sum << endl;
  return 0;
}
```

ข้อมูลในไฟล์ชื่อ number.dat:

```
5 6 7 33 4 0 19 1 4 9 24 36
```

ผลการทำงาน<u>ทางหน้าจอ</u>:

```
sum = 148
```

โค้ดตัวอย่างนี้จะอ่านเลขจำนวนเต็มจากไฟล์ที่ชื่อว่า number.dat แล้วบวกเลขในไฟล์นั้นทั้งหมดลงในตัวแปร sum จากนั้นจะเขียนค่าของ sum ออกทางหน้าจอ เมื่อเราใส่คำสั่ง fin >> value เข้าไปในเงื่อนไขของการลูป โปรแกรมจะอ่านเลขจำนวนเต็มจากไฟล์ number.dat เข้ามาเรื่อยๆ จนถึงเลข 36 ซึ่งเป็นตัวเลขสุดท้ายในไฟล์ หลังจากนั้นเมื่อโปรแกรมพยายามอ่านไฟล์มันก็จะอ่านไม่ได้เพราะข้อมูลหมดแล้ว คำสั่ง fin >> value ก็จะคืนค่ามาเป็นเท็จและทำให้ลูปหยุดทำงาน

ตัวอย่างการเขียนและอ่านไฟล์เพิ่มเติม

โค้ด:

```
int main()
{
   int value = 0;
   ifstream fin("number.dat");
   ofstream fout_odd("odd.dat");
   ofstream fout_even("even.dat");
   while (fin >> value)
   {
       if(value%2){
           fout_odd << value << " ";
       }
       else {
           fout_even << value << " ";
       }
   }
   cout << "Done!" << endl;
   return 0;
}
```

ข้อมูลในไฟล์ชื่อ number.dat:

```
12 11 20 15 35 10 69 71 23 80
```

ในตัวอย่างนี้ โปรแกรมเปิดไฟล์ 3 ไฟล์ คือเปิดเพื่ออ่านข้อมูล 1 ไฟล์ คือไฟล์ number.dat และเปิดเพื่อเขียนข้อมูล 2 ไฟล์ คือไฟล์ odd.dat และ even.data หลังจากนั้นโปรแกรมจะวนลูปเพื่ออ่านเลขจำนวนเต็มจากไฟล์ แล้วถ้าเลขที่อ่านเข้ามาเป็นเลขคี่ ก็จะบันทึกเลขนั้นลงในไฟล์ชื่อ odd.dat ไม่เช่นนั้น (นั่นคือถ้าเป็นเลขคู่) ก็จะบันทึกเลขนั้นลงในไฟล์ชื่อ even.dat เมื่ออ่านจบทั้งไฟล์แล้ว ก็จะแสดงผลออกทางหน้าจอว่า Done!

ผลการทำงานในไฟล์ odd.dat:

```
11 15 35 69 71 23
```

ผลการทำงานในไฟล์ even.dat:

```
12 20 10 80
```

ผลการทำงานทางหน้าจอ:

```
Done!
```

สรุปการทำงานกับไฟล์

1) #include <fstream> ทุกครั้ง ไม่ว่าจะอ่านหรือเขียนลงไฟล์

2) ต้องมีการเปิดไฟล์ก่อนทุกครั้ง โดยให้คำสั่งสร้างวัตถุสำหรับการอ่านหรือเขียนไฟล์ โดยสร้างวัตถุ 1 ชิ้นต่อการอ่านหรือเขียน 1 ไฟล์

3) เราไม่จำเป็นต้องอ่านจากไฟล์และเขียนลงไฟล์เท่านั้น เราสามารถอ่านจากแป้นพิมพ์แล้วเขียนลงไฟล์ หรืออ่านจากไฟล์แล้วเขียนออกทางหน้าจอก็ได้

สรุปสิ่งที่ควรได้จากบทเรียน

ส่วนของสตริง (string)

1) การประกาศสตริงและการให้ค่าในรูปแบบต่างๆ
2) เมธอด (method) หรือฟังก์ชัน (function) ที่ใช้กับสตริง
3) ตัวดำเนินการกับสตริง
    a. เราสามารถเปรียบเทียบสตริงโดยใช้เครื่องหมาย ==, !=, >, >=, <, และ <= ได้
    b. เราสามารถมองสตริงเป็นอาเรย์ของอักขระ และทำการเข้าถึงแต่ละอักขระในสตริงโดยใช้รูปแบบอาเรย์ได้

ส่วนของการทำงานกับไฟล์

1) ข้อแตกต่างและความเหมือนระหว่างการอ่านจากไฟล์และการอ่านผ่านทางแป้นพิมพ์
2) ข้อแตกต่างและความเหมือนระหว่างการเขียนลงไฟล์และการแสดงผลทางหน้าจอ
3) วิธีการอ่านจนจบไฟล์