บทที่ 3
การออกแบบโปรแกรมด้วยฟังก์ชันและคลาส

วัตถุประสงค์

1. เพื่อให้นักศึกษาทราบถึงวิธีการออกแบบโปรแกรมด้วยฟังก์ชันและคลาส
2. เพื่อให้นักศึกษาสามารถใช้งานฟังก์ชันในไลบรารีมาตรฐานได้
3. เพื่อให้นักศึกษาสามารถแก้ปัญหาสำหรับโปรแกรมที่มีความซับซ้อนได้
4. เพื่อให้นักศึกษาสามารถกำหนดและใช้งานฟังก์ชันที่ไม่มีอากิวเมนต์ได้ (Functions without Arguments)
5. เพื่อให้นักศึกษาสามารถกำหนดและใช้งานฟังก์ชันที่มีอากิวเมนต์ได้ (Functions with Input Arguments)
6. เพื่อให้นักศึกษาสามารถกำหนดและใช้งานฟังก์ชันที่มีอากิวเมนต์และมีการส่งผ่านผลลัพธ์กลับได้ (Functions with Input Arguments and Single Result)
7. เพื่อให้นักศึกษาสามารถกำหนดและใช้งานฟังก์ชันที่มีหลายอากิวเมนต์ได้ (Function with Multiple Arguments)
8. เพื่อให้นักศึกษาสามารถกำหนดและใช้งานฟังก์ชันที่มีการส่งผ่านค่าพารามิเตอร์ที่เป็นค่าหรืออ้างอิงได้ (Value and Reference Parameters)
9. เพื่อให้นักศึกษาสามารถแก้ปัญหากับโปรแกรมที่มีการใช้ชนิดข้อมูลร้อยความ(string)ได้

โปรแกรมเมอร์ที่ดีต้องมีวิธีการในการพัฒนาซอฟต์แวร์เพื่อช่วยในการแก้ปัญหาที่ดี โดยสามารถวิเคราะห์ความต้องการต่างๆของผู้ใช้ได้อย่างชัดเจนและครบถ้วน นำความต้องการต่างๆที่ได้รวบรวมไปออกแบบโปรแกรมเพื่อหาหนทางที่ดีที่สุดในการแก้ปัญหาต่างๆเหล่านั้นได้ แต่บางครั้งวิธีการในการแก้ปัญหาที่เลือกใช้ก่อให้เกิดความยุ่งยากและสับสน ถ้าเลือกวิธีการที่ไม่มี ประสิทธิภาพเพียงพอ จะส่งผลให้ยากต่อการเข้าใจและยากต่อการแก้ไข จึงเกิดวิธีการในการออกแบบขึ้นมาหลายเทคนิค สำหรับตำราเล่มนี้จะอธิบายถึงการออกแบบจากบนลงล่าง (top down design) ซึ่งเป็นเทคนิคที่มีการแบ่งปัญหาออกเป็นปัญหาย่อยๆให้แยกออกจากกัน ลักษณะบนลงล่าง โดยโปรแกรมจะประกอบด้วยโปรแกรมย่อยๆต่างๆที่มีการปฏิสัมพันธ์ต่อกันเป็นลำดับชั้น ผลดีของการออกแบบเทคนิคนี้คือ ปัญหาได้ถูกแบ่งเป็นปัญหาย่อยๆ และโปรแกรมย่อย 1 โปรแกรมจะแก้ปัญหาย่อยได้เพียง 1 ปัญหา ทำให้ง่ายต่อความเข้าใจและการแก้ไข

3.1 การสร้างโปรแกรม

โปรแกรมเมอร์เริ่มสร้างโปรแกรมจาก การทำความเข้าใจกับปัญหา วิเคราะห์ถึงข้อมูลนำเข้า ผลลัพธ์ที่ต้องการ กิจกรรมที่ต้องกระทำในโปรแกรมเป็นลำดับขั้นตอน โดยเปลี่ยนอัลกอริทึมให้เป็นโปรแกรมภาษาที่เหมาะสม ทำการทดสอบความถูกต้องของโปรแกรม เพื่อให้เข้าใจในกระบวนการสร้างโปรแกรมให้ดีมากยิ่งขึ้น มาดูตัวอย่างกรณีศึกษาดังต่อไปนี้

กรณีศึกษา :
การหาพื้นที่ของวงกลมและความยาวของเส้นรอบวง

ปัญหา ต้องการหาพื้นที่ของวงกลม และความยาวของเส้นรอบวง ของวงกลมใดๆ

วิเคราะห์ ต้องหาข้อมูลว่าอะไรคือข้อมูลเข้า อะไรคือผลลัพธ์ที่ต้องการ และต้องมีการประมวลผลอย่างไร ในที่นี้ข้อมูลเข้าคือความยาวของรัศมี ผลลัพธ์ที่ต้องการคือพื้นที่วงกลม และความยาวเส้นรอบวง สำหรับการประมวลผลต้องทราบสูตรในการคำนวณ ต่อจากนั้นนิยามข้อมูลต่างๆเหล่านี้เพื่อให้สามารถเก็บในหน่วยความจำของเครื่องคอมพิวเตอร์ได้

DATA REQUIREMENTS

Problem Constant

PI = 3.14159

Problem Input

float radius รัศมีของวงกลม

Problem Output

float area พื้นที่ของวงกลม

float circum ความยาวของเส้นรอบวง

Formular

พื้นที่ของวงกลม = π x radius x radius

ความยาวของเส้นรอบวง = 2 π x radius

ออกแบบ ขั้นตอนในการแก้ปัญหาแบ่งออกเป็น 4 ขั้นตอนใหญ่ ประกอบด้วย

ALGORITHM

1. รับความยาวรัศมี
2. คำนวณหาพื้นที่ของวงกลม
3. คำนวณหาความยาวของเส้นรอบวง
4. แสดงค่าพื้นที่ของวงกลม และความยาวของเส้นรอบวง ทางจอภาพ

พิจารณาแต่ละขั้นตอนว่าชัดเจนและสมบูรณ์หรือไม่ ปรากฏว่าในขั้นตอนที่ 2 และขั้นตอนที่ 3 ยังไม่สมบูรณ์ ดังนั้นต้องกำหนดสูตรในรายละเอียดเพื่อให้ชัดเจนขึ้นดังนี้

ALGORITHM WITH REFINEMENTS

1. รับความยาวรัศมี
2. คำนวณหาพื้นที่ของวงกลม
   2.1 area = PI x radius x radius
3. คำนวณหาความยาวของเส้นรอบวง
   3.1 circum = 2 x PI x radius
4. แสดงค่าพื้นที่ของวงกลม และความยาวของเส้นรอบวง ทางจอภาพ

ตัวอย่างที่ 3.1
โครงร่างของโปรแกรมหาพื้นที่และความยาวเส้นรอบวง

```
// Computes and displays the area and circumference of a circle
int main ( )
{
        const   float   PI = 3.14159;
        float  radius;                          // input : radius of circle
        float  area;                            // output : area of circle
        float  circum;                          // output : circumference of circle

        // Get the circle radius.
        // Compute area of circle.
        // Assign pi * radius * radius to area.
        // Compute circumference of circle.
        // Assign  2 * pi * radius * radius to circum
        //Display area and circumference.

        return 0;
}
```

เขียนโปรแกรม ทำการเปลี่ยนอัลกอริทึมเป็นภาษา C++ ดังนี้

ตัวอย่างที่ 3.2 หาพื้นที่ของวงกลมและความยาวของเส้นรอบวง

```
// File : circle.cpp
// Computes and displays the area and circumference of a circle
```

```
int main ( )
{
        const   float   PI = 3.14159;
        float  radius;                          // input : radius of circle
        float  area;                            // output : area of circle
        float  circum;                          // output : circumference of circle
        // Get the circle radius.
        cout <<  " Enter the circle radius : " ;
        cin  >> radius;
        // Compute area of circle.
        area = PI * radius * radius ;
        // Compute circumference of circle.
        circum =   2 * PI * radius ;
        // Display area and circumference.
        cout  <<  " The area of  the circle is " << area << endl;
        cout <<  " The circumference of the circle is "  << circum << endl;
        return 0;
}
```

ทดสอบ โดยป้อนความยาวรัศมีเท่ากับ 5.0 ทางแป้นพิมพ์ ผลลัพธ์ทางจอภาพเป็นดังนี้

```
Enter the circle radius : 5.0
The area of  the circle is 78.539749
The circumference of the circle is 31.415901
```

กรณีศึกษา :
การหาน้ำหนักรวมของกลุ่มวงแหวนสวมเกลียว

ปัญหา: ต้องการหาน้ำหนักของวงแหวนสวมเกลียว(flat washer)

วิเคราะห์: ต้องหาข้อมูลว่าอะไรคือข้อมูลเข้า อะไรคือผลลัพธ์ที่ต้องการ และต้องมีการประมวลผลอย่างไร ในที่นี้ข้อมูลเข้าคือความยาวของรัศมี ผลลัพธ์ที่ต้องการคือพื้นที่วงกลมและความยาวเส้นรอบวง สำหรับการประมวลผลต้องทราบสูตรในการคำนวณ ต่อจากนั้นนิยามข้อมูลต่างๆเหล่านี้เพื่อให้สามารถเก็บในหน่วยความจำของเครื่องคอมพิวเตอร์ได้

DATA REQUIREMENTS

Problem Constant

PI = 3.14159

Problem Inputs

float holeDiameter	// diameter of hole
float edgeDiameter	// diameter of outer edge
float thickness	//thickness of washer
float density	//density of material used
float quantity	//number of washers made

Problem Outputs

float weight	// weight of batch of washers

Program Variables

float holeRadius	//radius of hole
float edgeRadius	//radius of outer edge
float rimArea	//area of rim
float unitWeight	//weight of 1 washer

Relevant Formulas

area of circle = ¶ * radius^2

radius of circle = diameter/2

rim area = area of outer circle – area of hole

unit weight = rim area*thickness*density

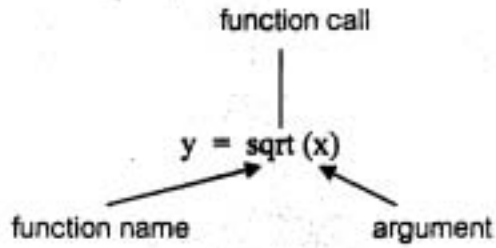
ตัวอย่างที่ 3.3 Washer program

```
// File : washers.cpp
// Computes the weight of a batch of flat washers.
#include <iostream>
using namespace  std;

int  main ( )
{
        const  float  PI = 3.14159
        float  holeDiameter ;          //input - diameter of hole
        float  edgeDiameter;           //input -  diameter of outer edge
        float thickness ;              //input - thickness of washer
        float density    ;             //input - density of material used
        float quantity ;               //input - number of washers made
        float weight;                  //output - weight of batch of washers
        float  holeRadius       ;      //radius of hole
        float  edgeRadius;             //radius of outer edge
        float rimArea;                 //area of rim
        float unitWeight;              //weight of 1 washer

        //Get the inner diameter , outer  diameter , and thickness.
```

```
    cout << " Inner diameter in centimeters : " ;
    cin >> holeDiameter ;
    cout << " Outer diameter in centimeters : " ;
    cin >> edgeDiameter;
    cout << " Thickness in centermeters : ";
    cin >> thickness ;

    // Get  the material  density and quantity manufactured.
    cout << " Material density  in grams per cubic centimeter ";
    cin >> density ;
    cout << " Quantity  in  batch  : ";
    cin >> quantity;

    //Compute the rim area .
    holeRadius = holeDiameter  / 2.0 ;
    edgeRadius = edgeDiameter / 2.0 ;
    rimArea = PI * edgeRadius * edgeRadius –  PI  * holeRadius * holeRadius;

    //Compute the weight of a flat washer.
    unitWeight = unitWeight * quantity;

    // Display the weight of the batch of washers.
    cout <<  " The expected weight of the batch is"
         << weight << " grams. "  << endl;
    return 0;
}
```

3.2 Library Function

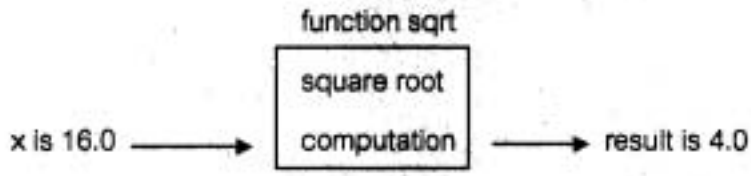
วัตถุประสงค์หลักของวิศวกรรมซอฟต์แวร์คือพัฒนาคำสั่งโปรแกรมให้ปราศจากความผิดพลาด วิธีการหนึ่งคือการนำคำสั่งที่เขียนและผ่านการทดสอบเรียบร้อยแล้วกลับมาใช้ใหม่ ซึ่ง C++ ได้เตรียมคลาสที่กำหนดไว้(predefined class) และฟังก์ชันในไลบรารีมาตรฐาน อาทิเช่น ไลบรารีมาตรฐาน cmath ซึ่งบรรจุฟังก์ชันทางคณิตศาสตร์มากมายเพื่อให้ผู้พัฒนาเลือกใช้ในการแก้ปัญหาทางด้านการคำนวณทางคณิตศาสตร์ เช่น ฟังก์ชัน sqrt ซึ่งเราใช้ในการคำนวณหาค่ารากที่สองของจำนวนตัวเลข



การทำงานจะมีการส่งผ่านอากิวเมนต์ x ไปยังฟังก์ชัน sqrt เมื่อถูกเรียกใช้ (function call) ถ้า x มีค่าเท่ากับ 16.0 การทำงานเป็นลำดับขั้นตอนดังนี้

1. x = 16.0 ดังนั้นฟังก์ชัน sqrt จะทำการคำนวณหาค่ารากที่สองของ 16.0 ซึ่งคือค่า 4.0
2. ซึ่งผลลัพธ์รองฟังก์ชันมีค่าเท่ากับ 4.0 ซึ่งจะมีผลให้ y มีค่าเท่ากับ 4.0

การทำงานของฟังก์ชันนี้เราเรียกว่า "black box" กล่าวคือมีการส่งผ่านค่าซึ่งเป็นข้อมูลเข้าไปยังฟังก์ชัน และมีการส่งผ่านค่าผลลัพธ์กลับมาโดยอัตโนมัติ ซึ่งผู้เรียกใช้ไม่ต้องทราบว่ามีวิธีการหรือกระบวนการทำงานอย่างไร ดังรูปที่ 3.1



รูปที่ 3.1 Function sqrt as a "black box"

ตัวอย่างที่ 3.4

แสดงการหาค่ารากที่สองของเลข 2 จำนวนใดๆ ซึ่งมีการเรียกใช้ฟังก์ชัน 3 เที่ยวดังนี้

```
answer = sqrt(first);
answer = sqrt(second) ;
answer = sqrt(first + second) ;
```

โดยการเรียกใช้ฟังก์ชัน sqrt นี้ต้องมีการกำหนดไลบรารีมาตรฐาน cmath ในส่วนเริ่มต้นของโปรแกรมด้วย #include <cmath> ดังตัวอย่างที่ 3.5

ตัวอย่างที่ 3.5 แสดงการใช้ฟังก์ชัน sqrt

```
// File: squareRoot.cpp
// Performs three square root computations
#include <cmath>                    // sqrt function
#include <iostream>                 // i/o functions
using namespace std;

int main ( )
{
    float first;                    // input : one of two data values
    float second;                   // input : second of two data values
    float answer;                   // output : a square root value

    // Get first number and display its square root.
    cout << "Enter the first number:  " ;
    cin >> first ;
    answer = sqrt ( first );
    cout  << "The square root of the first number is  " << answer << endl ;
```

```
    // Get second number and display its square root.
    cout << "Enter the second number : ";
    cin >> second;
    answer = sqrt (second);
    cout << "The square root of the second number is " << answer << endl;

    // Display the square root of the sum of first and second.
    answer = sqrt ( first + second );
    cout << "The square root of the sum of both numbers is " << andwer << endl;

    return 0;
}
```

```
Enter the first number: 9
The square root of the first number is 3
Enter the second number : 25
The square root of the second number is 5
The square root of the sum of both numbers is 5.83095
```

C++ Library Functions

ตารางที่ 3.1 แสดงรายละเอียดของฟังก์ชันถูกเรียกใช้งานเป็นประจำ รวมทั้งไลบรารีมาตรฐานที่ฟังก์ชันเหล่านี้บรรจุอยู่ ซึ่งโปรแกรมเมอร์ต้องกำหนดให้คอมไพเลอร์ได้ทราบก่อนเรียกใช้งาน

ตัวอย่างที่ 3.5 การใช้ฟังก์ชัน sqrt

เป็นการใช้ C++ โดยใช้ฟังก์ชัน sqrt และ pow เพื่อคำนวณหาค่าจากสมการดังนี้

$$root1 = (-b + \sqrt{b^2 - 4ac}) / 2a$$
$$root2 = (-b - \sqrt{b^2 - 4ac}) / 2a$$

กำหนดให้ $(b^2-4ac) >= 0$ เราสามารถเรียกใช้โดยเขียนคำสั่งกำหนดค่า root1 และ root2 ดังนี้

```
// Compute two roots, root1 & root2, for discriminant value > 0
disc = pow(b,2)-4.0*a*c;
root1 = (-b + sqrt(disc)) / (2.0*a);
root2 = (-b – sqrt(disc)) / (2.0*a);
```

ตัวอย่างที่ 3.6 การใช้ฟังก์ชัน cos

ถ้าเรารู้ความยาวของด้านสองด้านของสามเหลี่ยมใดๆ(b และ c) โดยมุมระหว่างด้าน b และ c มีค่าเท่ากับ 60 องศา เราสามารถคำนวณหาความยาวของด้านที่สาม(a) ได้จากสูตรดังนี้

$$a2 = b2 + c2 – 2bc(\cos(\alpha))$$

ซึ่งฟังก์ชัน cos นี้อยู่ในไลบรารีมาตรฐาน cmath ซึ่งถ้าเราทราบมุม α เป็นองศาต้องเปลี่ยนให้เป็นมุมเรเดียนเพื่อใช้ในการคำนวณ โดยคูณมุมองศาด้วยค่า ¶ /180 ดังนั้นเราสามารถหาความยาวของด้าน a ได้ดังนี้

```
a = sqrt(pow(b,2) + pow(c,2) – 2 * b * c * cos(alpha * pi / 180.0 )) ;
```

ตารางที่ 3.1 แสดงไลบรารีฟังก์ชันคณิตศาสตร์

Function	Standard Library	Argument	Result	Purpose:Example
abs(x)	<cstdlib>	int	int	หาค่าสัมบูรณ์ของตัวเลขจำนวนเต็ม เช่น abs(-5) มีค่าเท่ากับ 5
ceil(x)	<cmath>	double	double	หาค่าตัวเลขที่น้อยที่สุดแต่มากกว่า x เช่น ceil(45.23) มีค่าเท่ากับ 46.0
cos(x)	<cmath>	double (radians)	double	หาค่า cosine ของมุม x เช่น cos(0.0) มีค่าเท่ากับ 1.0
exp(x)	<cmath>	double	double	หาค่า e^x โดย e = 2.71828... เช่น exp(1.0) มีค่าเท่ากับ 2.71828
fabs(x)	<cmath>	double	double	หาค่าสัมบูรณ์ของเลขจำนวนจริง เช่น fabs(-8.43) มีค่าเท่ากับ 8.43
floor(x)	<cmath>	double	double	หาค่าที่มากที่สุดแต่น้อยกว่า x เช่น floor(45.23) มีค่าเท่ากับ 45.0
log(x)	<cmath>	double	double	หาค่าลอการิทึม ของ x โดย x>0.0 เช่น log(2.71828) มีค่าเท่ากับ 1.0
log10(x)	<cmath>	double	double	หาค่าลอการิทึมฐาน10 ของ x โดย x>0.0 เช่น log10(100.0) มีค่าเท่ากับ 2
pow(x,y)	<cmath>	double	double	หาค่า x^y เช่น pow(0.16,0.5) มีค่าเท่ากับ 0.4
sin(x)	<cmath>	double (radians)	double	หาค่า sine ของมุม x เช่น sin(1.5708) มีค่าเท่ากับ 1.0
sqrt(x)	<cmath>	double	double	หาค่ารากที่สองของ x โดย x >= 0.0 เช่น sqrt(2.25) มีค่าเท่ากับ 1.5
tan(x)	<cmath>	double (radians)	double	หาค่ามุม tangent ของมุม x เช่น tan(0.0) มีค่าเท่ากับ 0.0

3.3 Top-Down Design และ Structure Charts

กระบวนการแก้ปัญหาสำหรับโปรแกรมที่มีขนาดใหญ่และซับซ้อนนั้น โปรแกรมเมอร์ต้องแบ่งปัญหาทั้งหมดเป็นปัญหาย่อยๆ โดยพยายามให้เป็นอิสระต่อกันและแบ่งแยกย่อยลงไปเรื่อยๆ เป็นลำดับชั้น ซึ่งวิธีการนี้เราเรียกว่า top-down design เครื่องมือที่ช่วยในการแบ่งปัญหาต่างๆ นั้นตามกรณีศึกษาต่อไปนี้คือ structure chart ซึ่งจะช่วยเราในการกำหนดความสัมพันธ์ระหว่างปัญหาย่อยต่างๆ

กรณีศึกษา : การวาดภาพอย่างง่าย

ปัญหา ต้องการวาดภาพง่ายๆบนจอภาพ โดยวาดภาพบ้าน และ คน ตามรูปที่ 3.2

รูปที่ 3.2 แสดงภาพบ้าน และ คน

การวิเคราะห์ รูปบ้านนั้น เป็นการพิมพ์ รูปสามเหลี่ยมที่ไม่มีฐาน และพิมพ์ รูปของสี่เหลี่ยม ส่วน รูปคนนั้น พิมพ์ รูปวงกลม และพิมพ์ สามเหลี่ยมที่ไม่มีฐาน พิมพ์รูปเส้นตรง และพิมพ์รูปสามเหลี่ยมที่ไม่มีฐานอีกครั้งหนึ่ง ตามลำดับ

ดังนั้นเราสามารถวาดรูปที่ต้องการโดยแบ่งเป็นองค์ประกอบพื้นฐานของรูปร่างทั้งหมดได้ดังนี้

- a circle เป็นภาพวงกลม
- a base line เป็นเส้นตรง
- parallel lines เป็นเส้นขนาน
- intersecting lines เป็นสามเหลี่ยมที่ไม่มีฐาน

การออกแบบ สมมุติว่าเราต้องการสร้างรูปคน เราสามารถแบ่งปัญหานี้ออกเป็น 3 ปัญหาย่อยๆดังนี้

INITIAL ALGORITHM

1. วาดวงกลม
2. วาดสามเหลี่ยม
3. วาดสามเหลี่ยมไม่มีฐาน

ALGORITHM REFINEMENTS

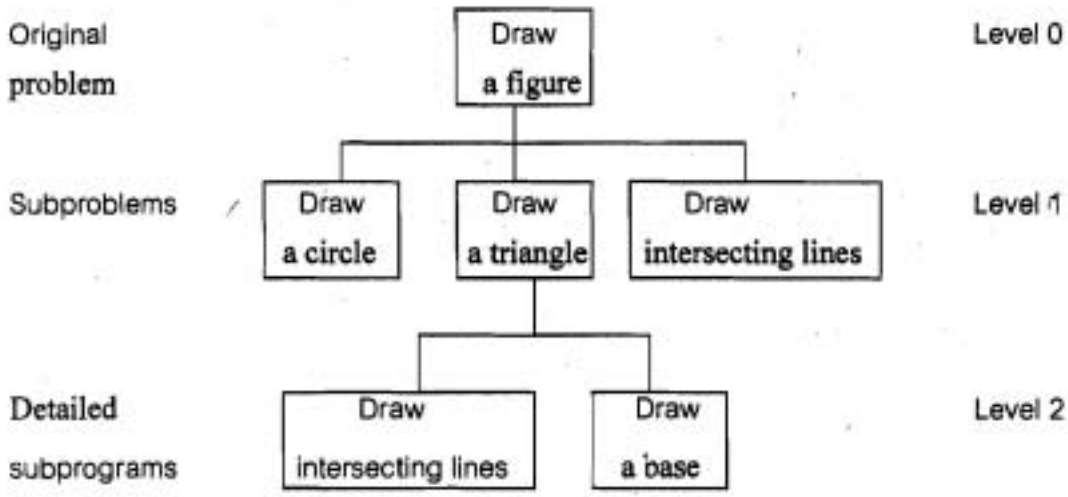
เนื่องจากรูปสามเหลี่ยมไม่มี เราสามารถกำหนดรูปนี้ได้จากองค์ประกอบอื่นๆ

Step 2 Refinement

2.1 วาดสามเหลี่ยมไม่มีฐาน

2.2 วาดเส้นตรง

ซึ่งเราสามารถใช้ structure chart แสดงความสัมพันธ์ของปัญหาหลักและปัญหาย่อยๆได้ดังรูปที่ 3.3 โดยกระบวนการของการแบ่งปัญหาเป็นลำดับขั้นเริ่มจากปัญหาหลักที่อยู่ในระดับบนสุดในที่นี้คือระดับ 0 และแบ่งปัญหาย่อยในการวาดภาพต่างๆเป็นระดับที่ 1 และ 2 ตามลำดับ



รูปที่ 3.3 Structure chart for drawing a stick figure

3.4 Function without Arguments

ในการแก้ปัญหานั้นโปรแกรมเมอร์สามารถใช้วิธีการออกแบบ top-down เพื่อพัฒนาโปรแกรม โดยแบ่งปัญหาหลักออกเป็นปัญหาย่อยๆ ในแต่ละปัญหาย่อยสามารถเขียนเป็นฟังก์ชันหรือโปรแกรมย่อยเพื่อแก้ปัญหาย่อยเหล่านั้น ภาษา C++ ได้เปิดโอกาสให้โปรแกรมเมอร์สามารถสร้างฟังก์ชันใหม่ที่ไม่มีการกำหนดไว้ในไลบรารี่มาตรฐานใดๆได้ เพื่อสะดวกต่อการใช้งานในบางลักษณะที่อาจเป็นการปฏิบัติงานที่กระทำซ้ำๆกันในโปรแกรม สำหรับหัวข้อนี้จะกล่าวถึงการเรียกใช้และกำหนดฟังก์ชันย่อยที่โปรแกรมเมอร์สามารถกำหนดขึ้น เพื่อใช้ในโปรแกรมในลักษณะที่ไม่มีอาร์กิวเมนต์และไม่มีการส่งผ่านค่าใดๆกลับ เป็นการวาดรูปตามต้องการ

ตัวอย่างที่ 3.7

Function prototypes and main function for drawing a stick figure

```
// Draws a stick figure (main function only)
#include <iostream>
using namespace std;
// Functions used ...
```

```
void drawCircle ( ) ;          / / Draws a circle
void drawTriangle ( );         / / Draws a triangle
void drawIntersect ( );        / / Draws intersecting lines
void drawBase ( );             / / Draws a horizontal line
int main ( )
{
    / / Draw a circle.
    drawCircle ( );
    / / Draw a triangle.
    drawTriangle ( );
    / / Draw intersecting lines.
    drawIntersect ( );
    return 0;
}
```

จากตัวอย่างในฟังก์ชันหลัก (main)นั้น มีการเรียกใช้ฟังก์ชันย่อย เช่น drawCircle( ) ; ซึ่งภายในวงเล็บไม่มีอาร์กิวเมนต์ใดๆและไม่มีการส่งผ่านค่าใดๆกลับมา

Function Call Statement(Function Without Arguments)
รูปแบบ: fname( );
ตัวอย่าง: drawCircle( ) ;

Function Prototypes

ฟังก์ชันใดๆที่กำหนดขึ้นมาใหม่โดยโปรแกรมเมอร์ ก่อนการเรียกใช้งานต้องมีการกำหนดให้ C++ รู้จักก่อน โดยการแทรกต้นแบบของฟังก์ชัน(Function prototypes) ที่กำหนดขึ้นใหม่นั้นก่อนฟังก์ชันหลัก ซึ่งต้นแบบฟังก์ชันจะทำหน้าที่บอกตัวแปลภาษา C++ ให้ทราบและรู้จักกับการปฏิบัติงานกับฟังก์ชันต่างๆเหล่านั้น จากตัวอย่างที่ 3.8 ต้นแบบของฟังก์ชันในการวาดวงกลมมีการเขียนคำสั่งได้ดังนี้ void drawCircle( ) ; โดยคำเฉพาะ void เป็นคำที่แสดงให้ทราบว่าฟังก์ชันนี้ไม่มีการส่งผ่านค่าใดๆกลับ

Function Prototype(Function Without Arguments)
รูปแบบ: ftype fname( ) ;
ตัวอย่าง: void drawCircle( ) ;

Function Definitions

แม้ว่าต้นแบบฟังก์ชันจะบอกให้ตัวแปลภาษาได้รู้จักถึงชื่อของฟังก์ชัน, อากิวเมนต์ที่ส่งผ่านและชนิดของผลลัพธ์แล้ว แต่การปฏิบัติงานของฟังก์ชันนั้นโปรแกรมเมอร์ต้องมีการเขียนคำสั่งการปฏิบัติงานของฟังก์ชันด้วยโดยแยกจากฟังก์ชันหลัก รูปแบบการกำหนดฟังก์ชันใหม่จะคล้ายๆกับฟังก์ชันหลักซึ่งประกอบด้วยหัวฟังก์ชันซึ่งต้องเหมือนกับที่กำหนดในต้นแบบฟังก์ชันแต่ไม่จบด้วยเครื่องหมายเซมิโคลอน ตามด้วยส่วนของการปฏิบัติงานของฟังก์ชันที่อยู่ภายในเครื่องหมายวงเล็บปีกกา ดังตัวอย่างที่ 3.8 ดังนี้

ตัวอย่างที่ 3.8 Function drawCircle

```
//Draws a circle
int drawCircle( )
```

```
{   cout << "  *  " << endl ;
    cout << "*    *" << endl ;
    cout << " **** " << endl ;
}
```

การใช้งานนั้นจะถูกเรียกใช้โดยคำสั่ง drawCircle( ) ; โดยการปฏิบัติงานภายในฟังก์ชันมีการประมวลผลคำสั่ง 3 คำสั่ง โดยพิมพ์รูปร่างวงกลมออกมาทางจอภาพและ กลับไปปฏิบัติงานในคำสั่งต่อไป

Function Definition (Function Without Arguments)

รูปแบบ : ftype fname( )
{
local declarations
executable statements
}

Function drawTriangle

```
// Draws a triangle
void drawTriangle ( )
{
    drawIntersect ( ) ;
    drawBase ( ) ;
}
```

การปฏิบัติงานภายในฟังก์ชันอาจมีการกำหนดตัวแปรที่ใช้เฉพาะภายในฟังก์ชัน หรือมีการเรียกฟังก์ชันอื่นๆมาใช้ภายในฟังก์ชันก็ได้ จากตัวอย่างที่ 3.9 นั้น แสดงให้เห็นถึงการเรียกใช้ฟังก์ชัน drawIntersect( ) และ drawBase ( ) เพื่อวาดรูปสามเหลี่ยมเป็นการเรียกฟังก์ชันซ้อนในฟังก์ชัน

เราสามารถเขียนคำสั่งโปรแกรมที่สมบูรณ์ในการเรียกใช้ฟังก์ชันต่างๆได้จากตัวอย่างที่ 3.9

ตัวอย่างที่ 3.9 Program to draw a stick figure

```
// File: stickFigure.cpp
// Drawa a stick figure
#include <iostream>
using namespace std;
// Functions used ...
void drawCircle ( ) ;              // Draws circle
void drawTriangle ( ) ;           // Draws a triangle
void drawIntersect ( ) ;          // Draws intersecting lines
void drawBase ( ) ;               // Draws a horizontal line
int main ( )
{
    // Draw a circle.
    drawCircle ( ) ;
    // Draw a triangle.
    drawTriangle ( ) ;
    // Draw intersecting lines.
    drawIntersect ( ) ;
    return 0 ;
}
// Draws a circle
void drawCircle ( )
{
    cout << "      *     " << endl;
    cout << "   *     *  " << endl;
    cout << "      * *    " << endl;
}                  // end drawCircle
```

```
// Draws a triangle
void drawTriangle ( )
{
    drawIntersect ( );
    drawBase( );
}   // end drawTriangle

//  Draws intersecting lines
void drawIntersect ( )
{
    cout << "        /\\        " << endl;
    cout << "       /  \\       " << endl;
    cout << "      /    \\      " << endl;
}   // end drawIntersect

// Draws a horizontal line
void drawBase ( )
{
    cout << " ______ " << endl;
}   // end drawBase
```

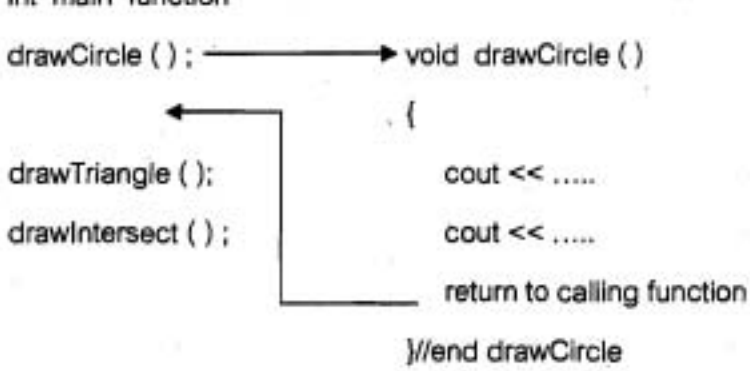
Order of Execution of Functions

ในการแปลภาษาโปรแกรมของภาษาC++นั้น ต้นแบบของฟังก์ชันที่สร้างขึ้นใหม่ต้องกำหนดให้คอมไพเลอร์แปลก่อนฟังก์ชันหลัก และเมื่อมีการเรียกใช้ฟังก์ชันต่างๆเหล่านั้นคอมไพเลอร์จะกระทำการแปลฟังก์ชันย่อยต่างๆเหล่านั้นเป็นลำดับ ซึ่งเมื่อมีการแปลคำสั่งจนถึงจุดจบของ

ฟังก์ชันเหล่านั้นตัวแปลภาษาจะแทรกคำสั่งภาษาเครื่องซึ่งเป็น transfer of control กลับไป ยังฟังก์ชันหลักเพื่อปฏิบัติงานคำสั่งต่างๆต่อไปจนจบโปรแกรม

จากรูปที่ 3.4 แสดงการทำงานเมื่อฟังก์ชันหลักมีการเรียกใช้ฟังก์ชัน drawCircle ( ) คอมพิวเตอร์จะจัดสรรเนื้อที่สำหรับตัวแปรที่มีการกำหนดในฟังก์ชันและเพื่อปฏิบัติงานในคำสั่งต่างๆภายในฟังก์ชัน และเมื่อมีปฏิบัติงานเป็นลำดับจนเสร็จสมบูรณ์แล้ว เนื้อที่ที่จัดสรรไว้นั้นจะถูกยกเลิกหรือถูกคืนให้กับหน่วยความจำส่วนกลาง ดังนั้นเนื้อที่ที่กำหนดขึ้นใหม่นี้จะเป็นเนื้อที่ที่ใช้งานเพียงชั่วคราวใช้เฉพาะเมื่อฟังก์ชันย่อยถูกปฏิบัติงานเท่านั้นเมื่อปฏิบัติงานเสร็จจะหายไป

โปรแกรมย่อยนี้มีประโยชน์สำหรับการแก้ปัญหาที่มีความซับซ้อน ในการพัฒนาโปรแกรมที่มีทีมงานหลายคนเราสามารถใช้เทคนิคนี้ช่วยในการจัดการโดยแจกจ่ายงานให้กับโปรแกรมเมอร์แต่ละคนให้ปฏิบัติงานโดยกำหนดถึงฟังก์ชันย่อยและการปฏิบัติงานให้รับผิดชอบ โดยโปรแกรมเมอร์แต่ละคนสามารถใช้ตัวแปรหรือการปฏิบัติงานที่เป็นอิสระต่อกัน และเมื่อฟังก์ชันย่อยต่างๆเหล่านั้นถูกพัฒนาขึ้นผ่านกระบวนการทดสอบความถูกต้องแล้ว เราสามารถนำฟังก์ชันต่างๆเหล่านี้กลับมาใช้ได้อีก เป็นการลดเวลาและค่าใช้จ่ายในการพัฒนาสำหรับโปรแกรมใหม่ๆ



รูปที่ 3.4 แสดงการเรียกใช้ระหว่างฟังก์ชันหลักและโปรแกรมย่อย

3.5 Functions with Input Arguments

อากิวเมนต์ของการเรียกใช้ฟังก์ชันนั้นเราใช้สำหรับส่งสารสนเทศไปให้ฟังก์ชัน ทำหน้าที่เป็นข้อมูลนำเข้าเพื่อนำไปประมวลผลในฟังก์ชัน ในบางกรณีเราใช้เป็นการส่งผ่านผลลัพธ์ของการ

ทำงานกลับไปให้กับฟังก์ชันหลักโดยเฉพาะในกรณีที่มีผลลัพธ์มากกว่า 1 ค่า สำหรับอาร์กิวเมนต์ที่ทำหน้าที่ส่งผ่านสารสนเทศไปยังฟังก์ชันเราเรียกว่า input arguments และอาร์กิวเมนต์ที่ทำหน้าที่ในการส่งผลลัพธ์กลับเรียกว่า output arguments แต่ถ้าการทำงานในฟังก์ชันมีผลลัพธ์เพียงค่าเดียวเราสามารถส่งผลลัพธ์กลับโดยใช้คำสั่ง return

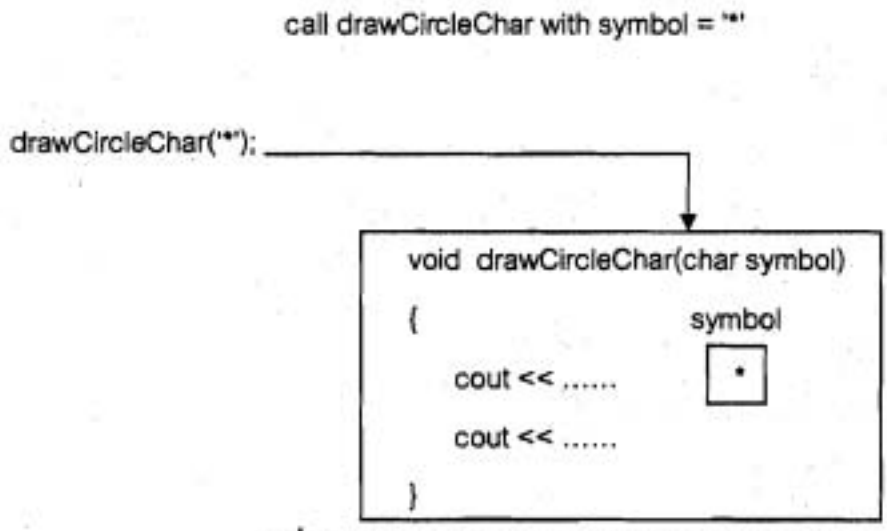
void Function with Input Arguments

ฟังก์ชันที่มีการกำหนดชนิดของผลลัพธ์ที่ส่งค่ากลับเป็น void นั้นจะไม่มีการ return หรือส่งค่าใดๆกลับไปยังจุดเรียกใช้

ตัวอย่างที่ 3.5

ฟังก์ชัน drawCircleChar ต่อไปนี้เป็นการปรับปรุงมาจาก drawCircle โดยเพิ่มอาร์กิวเมนต์ที่เป็นข้อมูลเข้าเพื่อส่งผ่านไปทำงานในฟังก์ชัน โดยกำหนดให้ส่งตัวอักขระ 1 ตัวอักษรไปยังฟังก์ชัน ดังนั้นการเรียกใช้จะถูกปรับเปลี่ยนเป็นดังนี้ drawCircleChar('*'); ข้อมูลนำเข้านั้นอาจเป็นอักขระใดๆที่ผู้ใช้ต้องการในที่นี้คืออักขระ '*' ดังนั้นการกำหนดฟังก์ชันเพื่อปฏิบัติงานต้องมีการปรับเปลี่ยนเช่นกัน โดยการทำงานของฟังก์ชันต้องมีการจัดสรรเนื้อที่ใหม่ซึ่งเป็นเนื้อที่ชั่วคราวในการจัดเก็บค่าที่ส่งมาเพื่อปฏิบัติงานในที่นี้จะจอง 1 byte เพื่อเก็บตัวอักขระที่ส่งมา โดยโปรแกรมเมอร์ต้องประกาศตัวแปรและชนิดของข้อมูลเพื่อใช้ในการรับค่าดังรูปที่ 3.5

```
// Draws a circle using the character specified by symbol
void drawCircleChar (char  symbol)
{   cout << "      " << symbol << endl ;
    cout << "   " << symbol << "      " << symbol << endl ;
    cout << "    " << symbol << " " << symbol <<  endl ;
}
// end  drawCircle
```



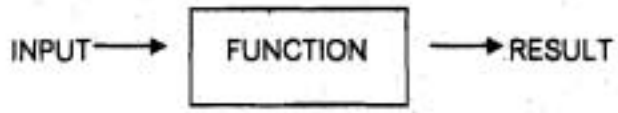
รูปที่ 3.5 แสดงภาพของการประมวลผลของ drawCircleChar('*')

นอกจากนี้การกำหนดต้นแบบเพื่อให้คอมไพเลอร์รู้จักนั้นจะมีรูปแบบที่เปลี่ยนไป ต้องมีการกำหนดถึงชนิดข้อมูลของอาร์กิวเมนต์ในการส่งผ่านให้คอมไพเลอร์ทราบด้วย ดังนี้

```
void  drawCircleChar (char) ;
```

Functions with Input Arguments and Single Result

จากหัวข้อที่ผ่านมาเป็นการสร้างฟังก์ชันที่เรียกไม่มีค่าในตัวเอง กล่าวคือไม่มีการส่งผ่านค่าผลลัพธ์ใดๆกลับมายังจุดเรียกใช้ ภาษา C++ เปิดโอกาสให้เราสามารถเขียนฟังก์ชันที่มีการส่งผ่านค่าไปยังฟังก์ชันและมีการส่งผ่านค่ากลับได้ ดังรูปที่ 3.6



รูปที่ 3.6 แสดงฟังก์ชันที่มีการส่งผ่านค่าไปยังฟังก์ชันและได้ผลลัพธ์ส่งกลับมา 1 ค่า

เพื่อให้เข้าใจได้ชัดเจนขึ้น เรามาดูตัวอย่างของการแก้ปัญหาของการหาพื้นที่และความยาวของเส้นรอบวงของวงกลมใดๆ ตามตัวอย่างที่ 3.9

ตัวอย่างที่ 3.9 Functions findCircum and findArea

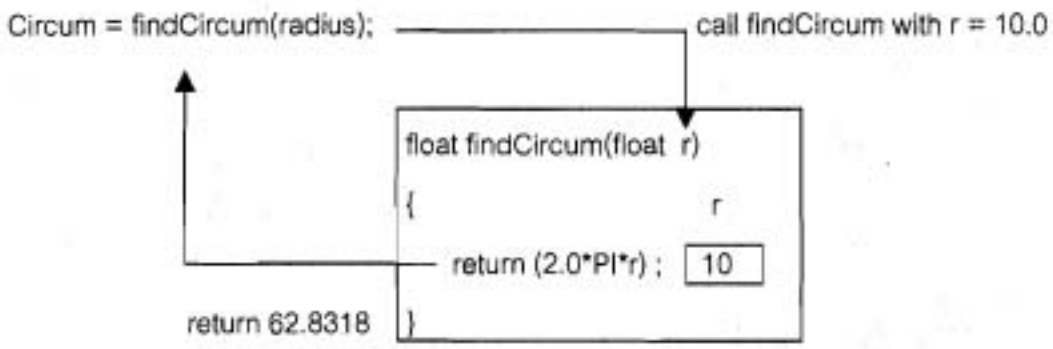
```
// Computes the circumference of a circle with radius r.
// Pre: r is defined and is > 0.
//          PI is a constant.
// Post: Returns
float findCircum (float r)
{
    return (2.0 * PI * r );
}

// Computes the area of a circle with radius r.
// Pre :  r is defined and is > 0.
//          PI is a constant.
//          Library cmath is included.
float findArea (float r )
{
    return  ( PI  *  pow ( r,  2 ) );
}
```

จากตัวอย่างที่ 3.9 นั้นในส่วนของหัวฟังก์ชันนั้นเริ่มต้นที่คำเฉพาะ float ซึ่งเป็นคำที่แสดงให้ทราบว่าผลลัพธ์ที่ส่งกลับไปยังจุดเรียกใช้เป็นเลขทศนิยม ซึ่งในส่วนของคำสั่งในการปฏิบัติงานมีคำสั่ง return ซึ่งส่งค่าจากการคำนวณ 2 * 3.14159 * r ไปยังจุดเรียกใช้ โดยส่งค่ากลับไป 1 ค่าเป็นเลขทศนิยม

ถ้ากำหนดให้การเรียกใช้เป็นดังนี้

```
radius  = 10.0 ;
circum = findCircum(radius) ;
```

ตัวแปร radius นี้เราเรียกว่า actual argument เป็นค่าจริงที่ส่งไปให้ฟังก์ชันในที่นี้คือ 10.0 ผลลัพธ์ของการทำงานนั้นเกิดจากการคำนวณค่า 2.0 * 3.14159 *10.0 ซึ่งคือค่า 62.8318 กลับมาให้ค่าแก่ตัวแปร circum ซึ่งการปฏิบัติงานนั้นมีขั้นตอนตามรูปที่ 3.7



รูปที่ 3.7 แสดงการประมวลผลของ circum = findCircum(radius)

ถ้ากำหนดให้การเรียกใช้เป็นดังนี้ area = findArea(radius) ; การปฏิบัติงานจะส่งค่า radius ในที่นี้คือ 10.0 ไปยังฟังก์ชัน findArea ซึ่งมีการคำนวณค่า 3.14159 * pow(r,2) ซึ่งการทำงานนี้มีการเรียกใช้ ฟังก์ชันมาตรฐาน cmath ของฟังก์ชันยกกำลัง ซึ่งผลลัพธ์ของการคำนวณคือ 3.14159 * 10.0 *10.0 ผลลัพธ์ของการทำงานที่ส่งกลับมีค่าเท่ากับ 314.59 ให้ค่าแก่ตัวแปร area ซึ่ง การใช้งานฟังก์ชันนี้ต้องมีการประกาศต้นแบบต่างๆไว้ก่อนฟังก์ชันหลัก ดังนี้

```
float  findArea(float);
float  findCircum(float);
```

ต้นแบบและการนิยามฟังก์ชันนั้นต้องกำหนดให้สอดคล้องและเหมือนกันทั้งชนิดข้อมูล และชื่อต่างๆ ต้องระมัดระวังด้วยว่าอักษรตัวใหญ่ตัวเล็กภาษา C++ จะถือว่าเป็นคนละตัวกัน ในส่วนของต้นแบบต้องสิ้นสุดด้วยเครื่องหมายเซมิโคลอน แต่ในส่วนของการนิยามฟังก์ชันจะไม่มี

Function Definition (Input Arguments and One Result)

รูปแบบ: // function interface comment

```
ftype  fname (formal-parameter-declaration-list)
{
    local variable declarations
    executable statements
}
```

ตัวอย่าง : // Finds the cube of its argument.

```
// Pre : n is defined .
int  cube (int n)
{
    return (n*n*n) ;
} // end cube
```

Function Prototype (With Parameters)

รูปแบบ : ftype fname (formal-parameter-type-list) ;

ตัวอย่าง : int cube(int) ;

Function Interface Comments

การกำหนดฟังก์ชันที่ดีต้องมีคำอธิบายถึงสารสนเทศต่างๆในฟังก์ชัน เราถือว่าเป็นเอกสารภายในโปรแกรม ซึ่งช่วยผู้พัฒนาและผู้บำรุงรักษาเกิดความเข้าใจได้ง่ายซึ่งก่อให้เกิดผลดีในการปรับปรุงแก้ไขหรือบำรุงรักษาในภายหลัง รวมทั้งการนำฟังก์ชันนี้กลับมาใช้ใหม่(reuse) คำอธิบายที่สำคัญเช่น

```
// Pre :  r is defined
```

เป็นคำอธิบายสารสนเทศ โดย Pre ย่อมาจาก precondition เป็นการบรรยายถึงเงื่อนไขที่เป็นจริงก่อนฟังก์ชันถูกเรียกหรือก่อนปฏิบัติงาน สำหรับ postcondition เป็นเงื่อนไขของการทำงานหลังจากฟังก์ชันประมวลผลเรียบร้อยแล้ว

Problem Inputs versus Input Parameters

การกำหนดฟังก์ชันนั้นมีโปรแกรมเมอร์มือใหม่หลายๆคนเข้าใจผิดซึ่งเขียนคำสั่งรับข้อมูลภายในฟังก์ชัน ข้อมูลเข้าที่เข้าสู่ฟังก์ชันนั้นต้องมีการส่งผ่านมาจากอาร์กิวเมนต์ที่กำหนด ต้องมีค่ามาก่อนแล้ว ความแตกต่างระหว่าง input กับ input parameters คือ input เป็นการรับข้อมูลหรือป้อนข้อมูลจากผู้ใช้โปรแกรมไปเก็บไว้ในตัวแปรที่กำหนด แต่ input parameters นั้น เป็นการส่งผ่านข้อมูลที่มีอยู่แล้วซึ่งมีการจัดเก็บไว้ในหน่วยความจำหรือตัวแปรใดๆ แต่ส่งผ่านเป็นข้อมูลเข้าไปยังฟังก์ชันที่กำหนดเพื่อให้ปฏิบัติงาน

Function with Multiple Arguments

ตัวอย่างที่ 3.10 แสดงถึงการส่งผ่านข้อมูลไปยังฟังก์ชันได้มากกว่า 1 ค่า

ตัวอย่างที่ 3.10 Function scale

```
/ / Multiplies its first argument by the power of 10
```

```
// specified by its second argument.
// Pre  : x and n are defined and library cmath is
//              included.
float scale ( float  x,  int  n )
{
    float  scaleFactor ;            // local variable
    scaleFactor = pow(10 , n) ;
    return  ( x *  scaleFactor ) ;
}
```

จากตัวอย่าง ถ้ามีการเรียกใช้ scale(2.5 , 2) จะมีการส่งผ่านค่า 250.0 กลับ แต่ถ้ามีการเรียกใช้ scale(2.0 , -2) จะมีการส่งผ่านค่า 0.025 กลับ

ตัวอย่างที่ 3.11 Testing function scale

```
// File testScale.cpp
// Tests function scale.
#include <iostream>
# include <cmath>
using namespace std;
// Function prototype
float scale (float,  int);
int main ( )
{
    float num1 ;
    int num2 ;
    // Get values for num1 and num2
    cout  <<  "Enter a real number :  " ;
```

```
    cin >> num1 ;
    cout << " Enter an integer: " ;
    cin >> num2 ;

    // Call scale and display result .
    cout << "Result of call to function scale is "
         << scale (num1 , num2 )                    // actual arguments
         << endl ;
    return 0 ;                                      // information flow
}
float scale (float  x,  int  n)                     // formal parameters
{
    float scaleFactor ;                             // local variable
    scaleFactor = pow(10 , n);
    return (x * scaleFactor);
}
```

```
Enter a real number : 2.5
Enter an integer : -2
Result of call to function scale is 0.025
```

สำหรับตัวอย่างที่ 3.11 เป็นโปรแกรมที่สมบูรณ์โดยฟังก์ชันหลักมีการเรียกใช้ฟังก์ชัน scale ความสัมพันธ์ระหว่าง actual parameter กับ formal parameter เป็นดังนี้

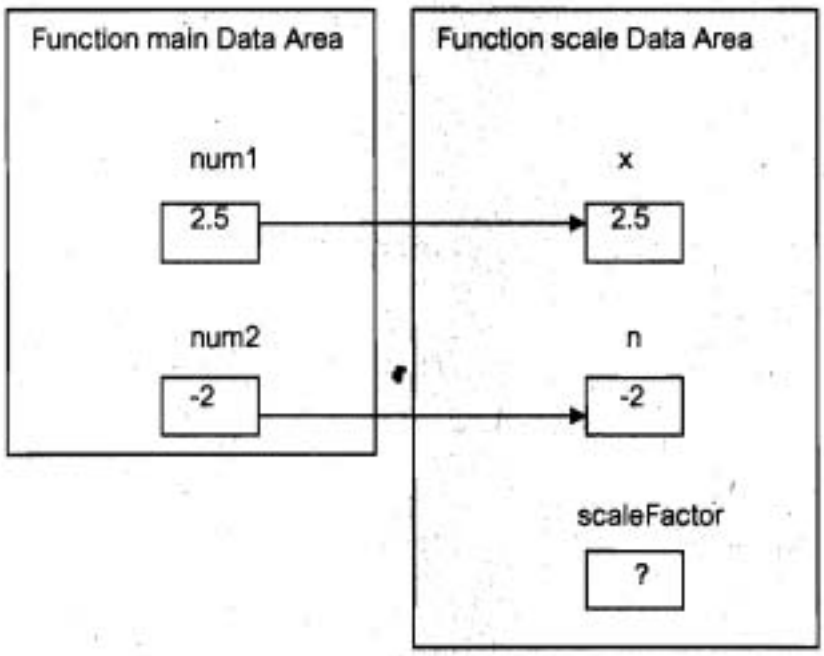
Actual Argument	corresponds to	Formal Parameter
num1		x
num2		n

ในกรณีที่มีอาร์กิวเมนต์หลายตัว เราต้องระมัดระวังในการกำหนดพารามิเตอร์เพื่อรับค่าซึ่งต้องมีความสอดคล้องกันและต้องมีลำดับตรงและชนิดของข้อมูลต่างๆเหมือนกัน สามารถสรุปได้ดังนี้

- จำนวนของอาร์กิวเมนต์จริงที่ใช้ในการส่งผ่านค่าไปยังฟังก์ชันต้องมีจำนวนเท่ากับพารามิเตอร์ที่กำหนดในต้นแบบของฟังก์ชัน
- ลำดับของอาร์กิวเมนต์ต้องสอดคล้องกับพารามิเตอร์
- ชนิดของข้อมูลของอาร์กิวเมนต์และพารามิเตอร์ต้องสอดคล้องกัน

The Function Data Area

เมื่อมีการเรียกใช้ฟังก์ชัน คอมพิวเตอร์จะมีการจัดสรรเนื้อที่ใหม่เป็นเนื้อที่ชั่วคราวในการเก็บค่าของ formal parameters และตัวแปรภายในฟังก์ชัน (local variables) และจะคืนเนื้อที่กลับไปยังหน่วยความจำส่วนกลางเมื่อฟังก์ชันกระทำงานจบลง และเนื้อที่นี้จะถูกจัดสรรขึ้นมาใหม่อีกเมื่อมีการเรียกใช้อีก รูปที่ 3.8 แสดงถึงการเรียกใช้ฟังก์ชัน scale ซึ่งมีการส่งผ่านค่า 2.5 และ -2 ไปยังฟังก์ชันเป็นลำดับ ดังนี้



รูปที่ 3.8 แสดงการจัดสรรเนื้อที่จากการเรียกใช้ฟังก์ชัน scale

3.6 Scope of Names

ขอบเขตของการใช้งานของชื่อและตัวแปรต่างๆในโปรแกรมนั้นในแต่ละส่วนจะมีการอ้างอิงที่แตกต่างกัน ตัวอย่างที่ 3.11 ตัวแปร num1 จะมีขอบเขตการใช้งานเฉพาะในส่วนของฟังก์ชันหลัก แต่สำหรับตัวอย่างที่ 3.12 นั้นตัวแปร MAX และ LIMIT เป็นตัวแปรคงที่ที่สามารถใช้งานได้ตลอดโปรแกรมและสามารถอ้างอิงในฟังก์ชันต่างๆได้อีกด้วย

ตัวอย่างที่ 3.12 Outline of program for studying scope of names

```
void one (int anArg,  double second) ;          // prototype 1
int  funTwo (int one, char anArg);              // prototype 2
const  int MAX = 950;
const  int LIMIT = 200;
int main ( )
{
    Int  localVar ;
    . . .
}   // end main
void one (int  anArg,  double second)           // header 1
{
    int oneLocal ;                              // local 1
    . . .
}   // end one
int  funTwo (int one,  char anArg)              // header 2
{
    Int localVar ;                              // local 2
    . . .
}   // end funTwo
```

ตารางที่ 3.2 แสดงขอบเขตการทำงานของชื่อต่างๆ ของ ตัวอย่างที่ 3.12

Name	Visible in One	Visible in funTwo	Visible in main
MAX	yes	yes	yes
LIMIT	yes	yes	yes
Main	yes	yes	yes
localVar(in main)	no	no	yes
one(the function)	yes	no	yes
anArg(int parameter)	yes	no	no
second	yes	no	no
oneLocal	yes	no	no
funTwo	yes	yes	yes
one(formal parameter)	no	yes	no
anArg(char parameter)	no	yes	no
localVar(in funTwo)	no	yes	no

3.7 Value and Reference Parameters

ในการแก้ปัญหาบางอย่างนั้น เราต้องการให้การทำงานของฟังก์ชันนั้นสามารถส่งผ่านข้อมูลได้มากกว่า 1 ค่า ซึ่งภาษา C++ ได้เตรียมการปฏิบัติการดังกล่าวดังต่อไปนี้

ตัวอย่างที่ 3.12 โปรแกรมหาค่าผลรวมและค่าเฉลี่ย

สมมุติว่าเราต้องการสร้างฟังก์ชันในการคำนวณหาค่าผลรวมและค่าเฉลี่ยของเลขจำนวนจริงใดๆ 2 จำนวน ซึ่งการทำงานของฟังก์ชันจะมีผลลัพธ์เกิดขึ้น 2 ค่าคือ ค่าผลรวม และค่าเฉลี่ย ส่วนข้อมูลเข้าคือเลขจำนวนจริงใดๆ 2 จำนวนที่ส่งให้กับฟังก์ชันนั่นเอง การแยกแยะพารามิเตอร์ต่างๆของฟังก์ชันนั้นเราใช้เครื่องหมาย &(ampersand) เป็นตัวกำหนด โดยพารามิเตอร์ตัวใดมีการระบุ & จะแสดงให้ทราบว่ามีการส่งผ่านแบบอ้างอิง โดยจะไม่กำหนดหรือจองเนื้อที่ใหม่ในการปฏิบัติงานในฟังก์ชัน แต่จะใช้เนื้อที่เดียวกับพารามิเตอร์ที่ส่งผ่านมาจากจุดเรียกใช้ ดังนั้น

พารามิเตอร์ที่อยู่ในส่วนหัวของฟังก์ชันจะทำหน้าที่เสมือนเป็นที่เก็บผลลัพธ์จากการทำงาน เราสามารถกำหนดชื่อฟังก์ชัน และการเรียกใช้ได้ดังนี้

```
computeSumAve(x ,y , sum ,mean) ;
```

เห็นได้ว่าการเรียกใช้ฟังก์ชัน computeSumAve นี้มีการส่งผ่านพารามิเตอร์ 4 ตัวด้วยกันโดย x, y , sum ,mean เป็น Actual Argument ซึ่ง x , y เป็นค่าของข้อมูลที่เป็นเลขจำนวนจริงใดๆซึ่งทำหน้าที่เป็นข้อมูลเข้าให้แก่ฟังก์ชัน โดยก่อนการเรียกใช้ค่า x ,y ต้องมีค่าเก็บอยู่ก่อนแล้ว แต่ sum และ mean ยังไม่มีค่าใดๆ แต่เมื่อมีการเรียกใช้ฟังก์ชันนี้ผลลัพธ์จากการทำงานจะให้ค่าแก่ sum และ mean

```
// File : computeSumAve.cpp
// Tests function computeSumAve.
#include <iostream>
using namespace std;
// Function prototype
void computeSumAve(float , float , float& , float&);
int main ()
{
  float x ,       // input – first number
        y ,       // input – second number
        sum ,     // output – their sum
        mean ;    // output – their average
   cout << "Enter 2 numbers: " ;
   cin >> x >> y ;

   // Compute sum and average of x and y
   computeSumAve(x , y , sum , mean);
```

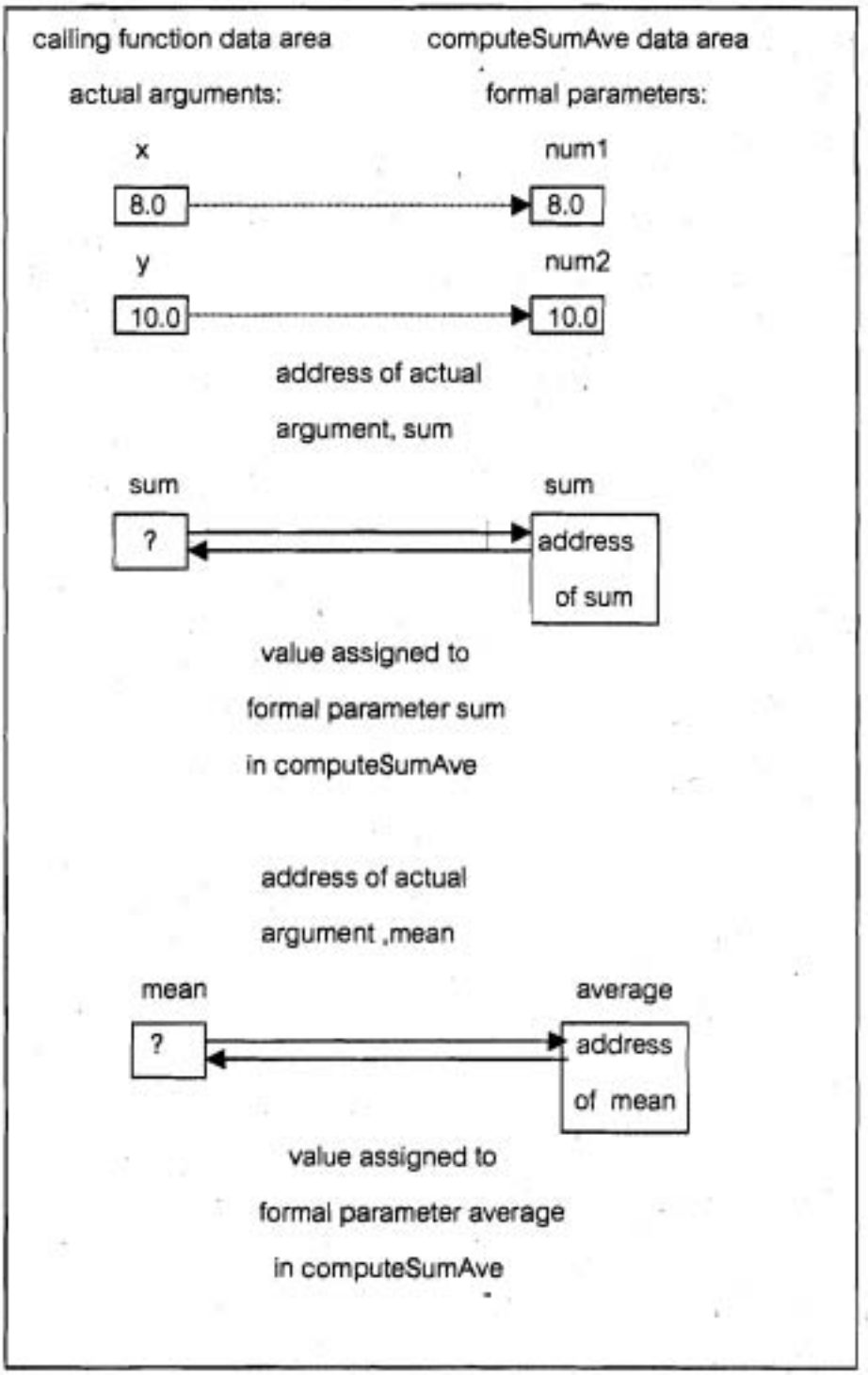
```
    // Display results
    cout << "Sum is " << sum << endl ;
    cout << "Average is " << mean << endl ;
    return 0 ;
}

// Computes the sum and average of num1 and num 2.
// Pre: num1 and num2 are assigned values.
// Post: The sum and average of num1 and num2 are
          computed and returned as function outputs.
void computeSumAve
    (float num1 ,              // IN – values used in
     float num2 ,              //      computation
     float& sum ,              // OUT – sum of num1 and num2
    float& average)            // OUT – average of num1 and num2
{
    sum = num1 + num2 ;
    average = sum / 2.0 ;
} // end computeSumAve
```

ทดสอบ

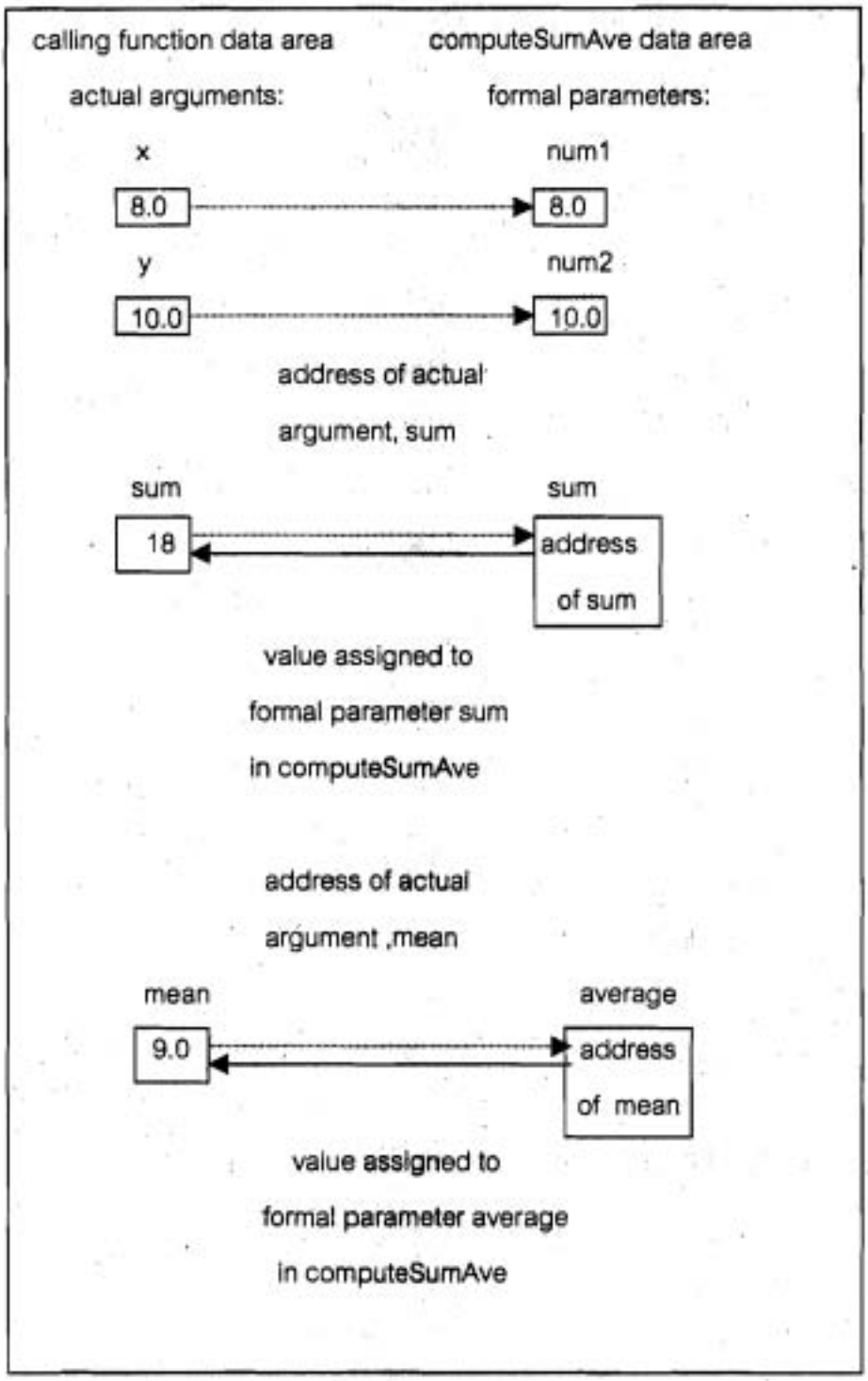
```
Enter 2 numbers : 8.0  10.0
Sum is 18
Average is 9
```

การทำงานของฟังก์ชันสามารถแสดงได้จากรูปที่ 3.9 **และ รูปที่** 3.10 **ดังนี้**

รูปที่ 3.9 แสดงพื้นที่ข้อมูลเมื่อมีการเรียกใช้ฟังก์ชัน computeSumAve ก่อนการปฏิบัติงาน



รูปที่ 3.10 แสดงพื้นที่ข้อมูลเมื่อมีการเรียกใช้ฟังก์ชัน computeSumAve หลังการปฏิบัติงาน



ในการทำงานคอมไพเลอร์จะตรวจสอบพารามิเตอร์ที่ได้ประกาศไว้ในฟังก์ชัน โดยจะกระทำงานตามที่ได้กำหนดไว้อย่างถูกต้อง สำหรับพารามิเตอร์ที่ทำหน้าที่เป็นข้อมูลเข้า (input) นั้น C++ ใช้ call – by – value โดยคัดลอกข้อมูลไปเก็บไว้ในหน่วยความจำที่กำหนด ไม่มีการเชื่อมโยงกันระหว่าง formal parameter และ actual parameter

สำหรับพารามิเตอร์ที่เป็นผลลัพธ์(output) C++ ใช้ call – by – reference ตัวแปลภาษาจะเก็บผลลัพธ์ที่เกิดจากการปฏิบัติงานในฟังก์ชัน ณ ตำแหน่งของตัวแปรจริง (actual variable) ที่สอดคล้องหรืออ้างอิงระหว่างกัน

3.8 Extending C++ through Classes: string and money

The string Class

โปรแกรมเมอร์สามารถกำหนดชนิดข้อมูลใหม่เพื่อใช้งานได้ในรูปของข้อความ โดยตัวกระทำที่สามารถปฏิบัติงานกับข้อมูลที่เป็นข้อความประกอบไปด้วย >> , << , = , + การประกาศตัวแปรที่เป็นข้อความสามารถกระทำได้ดังนี้

```
string   firstname , lastname ;     // inputs -first and last names
string   wholeName ;                // output- whole name
```

คอมพิวเตอร์จะทำการจัดสรรเนื้อที่สำหรับตัวแปรทั้งสามโดยเป็นข้อความว่าง(empty strings) แต่ถ้าเราต้องการให้ตัวแปรเก็บข้อความใดๆ เราสามารถประกาศได้ดังนี้

```
string  greeting = "Hello  " ;      // output – a greeting string
```

คอมพิวเตอร์จะจัดสรรเนื้อที่เก็บค่า "Hello " ในตัวแปรที่ชื่อว่า greeting สำหรับการอ่านข้อมูลเราสามารถใช้ cin >> firstName ; โดยผู้ใช้สามารถป้อนข้อความใดๆทางแป้นพิมพ์ คอมพิวเตอร์จะนำไปเก็บไว้ในตัวแปร firstName ตามลำดับ คำสั่งนี้จะไม่อ่านช่องว่าง และแสดงผลตัวแปรที่เป็นข้อความนี้ cout << greeting << wholeName << "!" << endl ;

เป็นคำสั่งในการแสดงข้อมูลในที่นี้เป็นข้อความจากตัวแปร greeting และข้อความของตัวแปร wholeName และเครื่องหมาย ! ออกทางจอภาพ ถ้าเราต้องการให้คอมพิวเตอร์อ่านช่องว่างด้วย เช่นการป้อนทั้งชื่อและนามสกุลไปเก็บไว้ในตัวแปรข้อความเพียงตัวแปรเดียวทำอย่างไร? เราสามารถใช้คำสั่งได้ดังนี้ getline(cin , firstName , '\n') ; เราสามารถนำข้อความต่างๆมาเชื่อมโยงกัน โดยใช้ตัวกระทำ + ดังนี้ wholeName = firstName + " " + lastName ;
ซึ่งผลจากการทำงาน ตัวแปร wholeName จะเปลี่ยนเป็นข้อความใหม่โดยนำข้อความของ firstName เชื่อมกับช่องว่าง 1 ช่อง ตามด้วยข้อความจากตัวแปร lastName จะเห็นได้ว่าตัวกระทำ + นี้สามารถปฏิบัติงานได้หลายลักษณะขึ้นอยู่กับชนิดของข้อมูล ถ้าเป็นตัวเลขเครื่องหมายนี้จะกระทำการบวก แต่ถ้าข้อมูลเป็นข้อความจะนำข้อความมาเชื่อมโยงกัน เราเรียกการปฏิบัติงานที่แตกต่างกันนี้ว่า Operator Overloading

ตัวอย่างที่ 3.13 Illustating string operations

```
// File: stringOperations . cpp
// Illustrates string operations
# include <iostream>
# include <string>
using namespace std;
int main ( )
{
    string firstName, lastName;// inputs – first and last names
    string wholeName;          // output - whole name
    string greeting = "Hello " ; // output – a greeting string

    // Read first and last names.
    cout << "Enter your first name : " ;
    cin >> firstName ;
    cout << "Enter your last name : " ;
```

```
    cin >> lastName;
    // Join names in whole name
    wholeName = firstName + " " + lastName;
    // Display results
    cout << greeting << wholeName << '!' << endl ;
    cout << "You have " << (wholeName . length ( ) - 1 )
         << " letters in your name. " << endl ;
    // Display initials
    cout << "Your initials are " << firstName . at (0)
         << lastName . at (0) << endl ;
    return 0;
}
```

```
Enter your first name : Caryn
Enter your last name : Jackson
Hello Caryn Jackson !
You have 12 letters in your name.
Your initials are CJ
```

จากตัวอย่างที่ 3.13 มีการเรียกใช้ฟังก์ชัน length และ at จาก string class เราใช้จุด(.) เพื่ออ้างอิงในการส่งผ่านค่าอาร์กิวเมนต์ต่างๆไปให้ฟังก์ชัน เช่น wholeName.length( ) โดยการเรียกใช้นี้ไม่มีการส่งผ่านค่าอาร์กิวเมนต์ใดๆ แต่การทำงานของฟังก์ชันจะส่งผ่านค่า 13 กลับมา ถ้าสมมุติว่า wholeName เก็บค่า "Caryn Jackson" ในปัจจุบัน ถ้าเราต้องการพิมพ์เฉพาะตัวอักษรตัวแรกของชื่อและนามสกุล ในการทำงานเราจะระบุตำแหน่งที่ตำแหน่ง 0 เนื่องจากภาษา C++ จะเริ่มเก็บค่าข้อความตั้งแต่ตำแหน่ง 0 เป็นต้นไป

Dot Notation
รูปแบบ : object.function-call **ตัวอย่าง** : firstName.at(0)

ถ้าเรากำหนดให้ wholeName = "Caryn Jackson" ; firstName = "Jackson"
ถ้ามีการเรียกใช้ wholeName.find(firstName) ; ผลของการทำงานจะมีค่าเท่ากับ 6 เนื่องจากตำแหน่งแรกจะต้องเริ่มต้นด้วย 0 เสมอ

ถ้าเราต้องการแทรกข้อความใหม่ในตำแหน่งเริ่มต้น และในตำแหน่งที่ 10 ทำอย่างไร ?

```
wholeName.insert(0 , "Ms. " );     // Change to
                                   // "Ms. Caryn Jackson"
wholeName.insert(10 , "Heather  " ) ; // Change to
                                   // "Ms. Caryn Heather Jackson"
```

ถ้าเราต้องการเปลี่ยนแปลงข้อความโดยปรับเปลี่ยนชื่อกลางเป็น "Amy" ทำอย่างไร ?

```
wholeName.replace(10 , 7 , "Amy") ; // Change to
                                   // "Ms. Caryn Amy Jackson"
```

ถ้าเราต้องการลบชื่อกลางทำอย่างไร ?

```
wholeName.erase(10, 4) ;           // Change back to
                                   // "Ms. Caryn Jackson"
```

นอกจากนี้เราสามารถดึงข้อมูลที่ต้องการจากข้อความใดๆให้ค่าแก่ตัวแปรได้ ดังนี้

```
title.assign(wholeName,0,3); // Store "Ms." In title
```

ตารางที่ 3.3 เป็นฟังก์ชันต่าง ๆที่บรรจุใน string Class

Function	Purpose
getline(cin,aString,'\n')	เป็นการดึงบรรทัดแรกจากสายของข้อมูลที่ผู้ใช้ป้อนเก็บในตัวแปร aString
aString.length()	ส่งผ่านความยาวของตัวอักษรที่เก็บในตัวแปร aString กลับมายังจุดเรียกใช้
aString.at(i)	ส่งผ่านตัวอักขระ ณ ตำแหน่งที่ I ของตัวแปร aString กลับมายังจุดเรียกใช้
aString.find(target)	ส่งค่าตำแหน่งเริ่มต้นของข้อความ target มายังจุดเรียกใช้แต่ถ้าไม่สามารถค้นหาพบจะส่งผ่านค่าที่น้อยกว่า 0 หรือค่าที่มีค่ามากกว่าความยาวของ target ส่งกลับมายังจุดเรียกใช้
aString.insert(start,newString)	เป็นคำสั่งในการแทรกข้อความใหม่ (newString) ณ ตำแหน่ง start เป็นจุดเริ่มต้น
aString.replace(start,count,newString)	เป็นคำสั่งในการแทนที่ข้อความโดยแทนที่ข้อความใหม่ (newString) เป็นจำนวน count ตัว ณ ตำแหน่งเริ่มต้น (start)
aString.erase(start,count)	ลบข้อความเป็นจำนวน count ตัวอักษรโดยเริ่มที่ตำแหน่ง start เป็นตำแหน่งแรก
aString.assign(oldString,start,count)	เป็นการดึงข้อมูลจาก oldStart เป็นจำนวน count โดยเริ่มจากตำแหน่ง start นำข้อความนี้ให้ค่าแก่ตัวแปร aString

สรุป

1. ขั้นตอนในการสร้างโปรแกรม เริ่มจากการทำความเข้าใจกับปัญหา วิเคราะห์ถึงข้อมูลเข้า ผลลัพธ์ที่ต้องการ และกิจกรรมที่ต้องกระทำในโปรแกรมเป็นลำดับขั้นตอน และเปลี่ยนอัลกอริทึมให้เป็นโปรแกรมภาษาและทดสอบความถูกต้อง
2. ภาษาC++ ได้มีการรวบรวมฟังก์ชันที่มีการใช้งานบ่อยๆในโปรแกรมรวบรวมไว้ในคลาสต่างๆ เช่น sqrt(x) เป็นฟังก์ชันที่ใช้ในการหาค่ารากที่สองของ x ที่อยู่ในคลาส cmath เป็นต้น
3. ฟังก์ชันที่ผู้ใช้กำหนดขึ้นใหม่นั้น ต้องมีการกำหนดต้นแบบฟังก์ชัน (Function Prototypes) ไว้ก่อนการเรียกใช้ฟังก์ชันนั้น
4. การปฏิบัติงานภายในฟังก์ชัน โปรแกรมเมอร์อาจมีการกำหนดตัวแปรที่ใช้เฉพาะภายในฟังก์ชัน หรือมีการเรียกฟังก์ชันอื่นๆมาปฏิบัติงานก็ได้
5. อาร์กิวเมนต์ของฟังก์ชันนั้นถ้าทำหน้าที่ส่งผ่านข้อมูลไปยังฟังก์ชันเรียกว่า input arguments ถ้าทำหน้าที่ในการส่งผลลัพธ์กลับ เรียกว่า output arguments
6. ฟังก์ชันที่มีการกำหนดชนิดผลลัพธ์เป็น void เป็นฟังก์ชันที่ไม่มีการส่งผลลัพธ์กลับ
7. เมื่อมีการเรียกใช้ฟังก์ชัน คอมพิวเตอร์จะมีการจัดสรรเนื้อที่ใหม่ชั่วคราวในการเก็บค่าที่ส่งมา และจะยกเลิกเนื้อที่นี้เมื่อการทำงานของฟังก์ชันสิ้นสุดลง
8. Operator Overloading หมายถึงตัวกระทำที่สามารถปฏิบัติงานได้หลายลักษณะขึ้นอยู่กับชนิดของข้อมูล เช่น ถ้านำ 3 + 4 ผลลัพธ์มีค่าเท่ากับ 7 แต่ถ้านำ '3' + '4' ผลลัพธ์มีค่าเท่ากับ "34" เป็นต้น

แบบฝึกหัด

1. จงสร้างฟังก์ชันในการวาดรูปต่อไปนี้

2. จงสร้างฟังก์ชัน average ซึ่งมีการเรียกใช้ได้ดังนี้

```
x = average( 2 , 6.5 ) + 3.0 ;
```

การทำงานจะมีการส่งผ่านค่า 2 จำนวน โดยจำนวนแรกเป็นแลขจำนวนเต็ม จำนวนที่สอง เป็นลรทศนิยมไปยังฟังก์ชัน และส่งผ่านค่าเฉลี่ยกลับมายังจุดเรียกใช้

3. จงหาผลลัพธ์รองคำสั่งต่อไปนี้

```
string flower = "rose" ;
flower = flower + " of Sharon" ;
cout << flower.at(0) << flower.at(8) << endl ;
cout << flower.find("s") << " " << flower.find("S") << endl;
flower.replace(5 , 2 , "from" ) ;
flower.erase( 0 , 4) ;
flower.insert(0 "thorn");
```

4. จงเขียนโปรแกรมภาษา C++ เพื่อหาค่าจากสมการต่อไปนี้

$$Y = (e^{n \ln b})^2$$

5. กำหนดให้ flower = "rose of Sharon" ;

5.1 จงเขียนคำสั่งที่แสดงผลเป็น "rS"

5.2 จงเขียนคำสั่งที่เปลี่ยนค่า flower เป็น "rose from Thailand"

5.3 จงเขียนคำสั่งที่เปลี่ยนค่า flower เป็น " from Thailand"

6. จงเขียนโปรแกรมรับเลขจำนวนจริง ใดๆ และปัดให้มีจำนวนทศนิยมเพียง 2 ตำแหน่ง

   INPUT : 32.4851

   OUTPUT : 32.49

   INPUT : 32.4431

   OUTPUT : 32.44

7. จงเขียนโปรแกรมคิดราคาส่วนลดและภาษีของการซื้อสินค้า โดยป้อนราคาสินค้าทางแป้นพิมพ์ นำมาคิดส่วนลด 5 % และ ภาษี 7 %

8. จงเขียนโปรแกรมในการรับความลึก(depth)หน่วยเป็นกิโลเมตร จากพื้นผิวโลก นำมาหาค่าของอุณหภูมิเป็นองศาเซลเซียส และองศาฟาเรนไฮน์

   Celsius = 10(depth) + 20

   Fahrenheit = 1.8(Celsius) + 32

9. เขียนโปรแกรมรับข้อความหนึ่งซึ่งบรรจุคำใดๆ4คำแยกคำด้วยอักขระ * เช่น A*book*is*good จงแยกคำออกจากกันและพิมพ์คำย้อนกลับจากหลังไปหน้า ดังนั้นผลลัพธ์พิมพ์ good*is*book*A ทางจอภาพ

10. จงเขียนโปรแกรมรับระยะทางเป็นกิโลเมตรใดๆ จำนวนนาที และจำนวนวินาที ของนักวิ่งใดๆที่จับเวลา จงหาว่านักวิ่งคนนี้มีอัตร่าการวิ่งความเร็วกี่ฟุตต่อวินาที โดย 1 กิโลเมตรท่ากับ 3281 ฟุต โดยเขียนเป็นฟังก์ชัน