โครงสร้างข้อมูล

ในการดำเนินชีวิตของคนปัจจุบัน ต้องเกี่ยวข้องกับข้อมูลอยู่เสมอ ๆ ลักษณะของข้อมูลที่พบเห็น มักเป็นข้อมูลเป็นกลุ่มของข้อมูล เช่น เมื่อพิจารณาข้อมูลของคนหนึ่งคน อาจจะพิจารณาคุณสมบัติใน เรื่องของชื่อ สกุล น้ำหนัก ความสูง อายุ หรือเมื่อพิจารณารถยนต์หนึ่งคันอาจจะพิจารณาถึง เป็นรถยนต์ที่ผลิตจากประเทศใด ยี่ห้อ รุ่น ขนาดเครื่องยนต์ สี เป็นต้น จะเห็นว่าเวลาที่พบเห็นข้อมูลหนึ่ง ๆ ข้อมูลเหล่านั้นมักจะประกอบด้วยคุณสมบัติหรือข้อมูลย่อยเสมอ กรณีของข้อมูลโครงสร้างก็เป็นการพิจารณาข้อมูลในลักษณะเดียวกัน การประกาศประเภทข้อมูลโครงสร้างขึ้นมาชนิดหนึ่ง จะต้องพิจารณาว่าข้อมูลโครงสร้างนั้นประกอบไปด้วยคุณสมบัติ อะไรบ้าง และจะต้องมีการประกาศคุณสมบัติของข้อมูลโครงสร้างด้วยเสมอ

ข้อมูลที่ใช้ในการเขียนโปรแกรมตามที่กล่าวมาในบทก่อนหน้านี้ได้นำเสนอ ข้อมูลเป็นแบบอาร์เรย์(ศรัณย์ อินทโกสุม, 2539) ซึ่งหมายถึง ชุดข้อมูลที่เป็นข้อมูลชนิดเดียวกันเก็บไว้ตัวแปรหนึ่งตัว นอกจากตัวแปรชุดแบบอาร์เรย์แล้ว ภาษาซียังมีชนิดข้อมูลที่ประกอบขึ้นจากข้อมูลประเภทพื้นฐานอีกประเภทหนึ่ง (Steven และ Lutfar, 2006) (Brian W. K., Online) ข้อมูลแบบโครงสร้าง (Structure) และข้อมูลแบบยูเนียน (Union) เป็นข้อมูลที่สามารถนำข้อมูลชนิดที่แตกต่างกันมาเก็บไว้เป็นกลุ่มเดียวกัน เช่น ข้อมูลของพนักงานในองค์กร มักประกอบด้วยข้อมูลพื้นฐานคือ รหัสพนักงาน ชื่อ สกุล ที่อยู่ เบอร์โทรศัพท์ บัญชีธนาคาร เป็นต้น

การประกาศโครงสร้างข้อมูล

```
struct    <ชื่อโครงสร้าง> {
                <สมาชิก 1> ;
                <สมาชิก 2> ;
                …
                <สมาชิก 3> ;
          } ;
```

struct เป็นคำสั่งใช้ในการประกาศให้ทราบว่าเป็นโครงสร้าง โดยมีชื่อของโครงสร้างเป็นตัวอ้างอิงในการเรียกใช้ต่อไป และการประกาศชื่อของโครงสร้าง ภายในโครงสร้างจะต้องประกอบด้วย สมาชิก ซึ่งจำนวนสมาชิกจะเป็นตัวแปรชนิดใด จำนวนเท่าใด ดังตัวอย่าง ในการประกาศโครงสร้างของการเก็บข้อมูลประชากรอาจประกอบด้วยสมาชิกของโครงสร้างคือ ชื่อ สกุล น้ำหนัก ความสูง อายุ สามารถประกาศได้ดังนี้

```
struct   population {
           char name[8];
           char surname[10];
           float weight;
           float height;
           int age;
} ;
```

จะพบว่าข้อมูลโครงสร้างเป็นการวางกรอบสมาชิกของข้อมูลที่เราพบเห็นให้เป็นระเบียบเป็นกลุ่มข้อมูลเดียวกันเพื่อ ให้สามารถอ้างอิงได้ง่ายขึ้น

รูปแบบการประกาศตัวแปรโครงสร้าง

การประกาศตัวแปรโครงสร้างสามารถทำได้หลากหลายรูปแบบ ซึ่งสามารถสรุปตัวอย่างให้เห็น 4 รูปแบบ ดังนี้ คือ

รูปแบบที่ 1

```
struct {
        int x;
        int y;
} p1, p2;
```

รูปแบบที่ 2

```
struct data {
        int x;
        int y;
} p1, p2;
```

รูปแบบที่ 3

```
struct data {
        int x;
        int y;
};
typedef struct data Point;
```

รูปแบบที่ 4

```
typedef struct {
        int x;
        int y;
} data;
```

จากรูปแบบการประกาศตัวแปรโครงสร้างด้านบน สามารถสรุปความแตกต่างได้คือ

รูปแบบที่ 1 เป็นการประกาศรูปแบบโครงสร้าง พร้อมกับการประกาศตัวแปรสำหรับโครงสร้าง จำนวน 2 ตัว คือ p1 และ p2

รูปแบบที่ 2 เป็นการประกาศรูปแบบโครงสร้างและตั้งชื่อรูปแบบนี้ว่า data พร้อมกับทำการประกาศตัวแปรสำหรับโครงสร้างนี้ 2 ตัว คือ p1 และ p2 การประกาศโครงสร้างในรูปแบบนี้สามารถนำโครงสร้างนี้ไปใช้สำหรับการประกาศตัวแปรใหม่ที่มีโครงสร้างเหมือนกับ p1 และ p2 ได้โดยไม่ต้องประกาศโครงสร้างนี้ใหม่ เช่น ต้องการประกาศตัวแปร p3 ซึ่งมีโครงสร้างรูปแบบเดียวกับ p1 และ p2 สามารถทำได้โดยเพิ่มคำสั่งต่อไปนี้

```
struct data p3;
```

โดยวางคำสั่งนี้ไว้ที่ตำแหน่งหลังคำสั่งประกาศโครงสร้าง data

ส่วนรูปแบบที่ 3 และ 4 เป็นการกำหนดชนิดข้อมูลใหม่โดยใช้คำสั่ง typedef เพื่อให้สามารถอ้างถึงรูปแบบโครงสร้างได้สั้นขึ้น จากรปู แบบที่ 2 เมื่อต้องการประกาศตัวแปรใหม่ต้องอ้างถึงโครงสร้างด้วยคำสั่ง struct data point1; หากใช้การประกาศโครงสร้างในรูปแบบที่ 3 หรือ 4 สามารถประกาศตัวแปรใหม่ด้วยคำสั่ง data point1; ได้ทันที

การประกาศค่าเริ่มต้นให้กับสมาชิกของตัวแปรโครงสร้าง

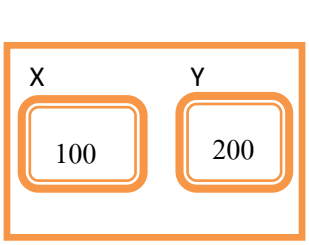
ในการประกาศตัวแปรพื้นฐานเราสามารถกำหนดค่าเริ่มต้นให้กับตัวแปรนั้นได้ ตัวอย่างเช่น ต้องการประกาศตัวแปร x มีชนิดเป็น int ให้มีค่าเริ่มต้นเท่ากับ 3 สามารถทำได้ด้วยคำสั่ง

```
int x =3;
```

สำหรับตัวแปรโครงสร้าง สามารถกำหนดค่าเริ่มต้นให้กับสมาชิกของตัวแปรโครงสร้างนั้นได้เช่นกัน ตัวอย่าง เช่น ถ้าต้องการสร้างตัวแปรโครงสร้างชื่อ pt ซึ่งเป็นตัวแปรที่มีรูปแบบโครงสร้าง data สามารถทำได้ดังนี้

```
data pt = { 100, 200 };
```

ค่า x ของตัวแปรโครงสร้าง pt จะมีค่าเท่ากับ 100 ค่า y จะมีค่าเท่ากับ 200 สามารถจำลองการเก็บข้อมูลตัวแปรโครงสร้าง pt รูปที่ 10.1



รูปที่ 10.1 แสดงแบบจำลองของตัวแปรโครงสร้าง

การอ้างถึงสมาชิกของตัวแปรโครงสร้าง

การอ้างถึงสมาชิกภายใน struct สามารถทำได้โดยการใช้คำสั่ง . (จุด) เป็นตัวอ้างอิงถึงสมาชิกภายในโครงสร้าง ดังในรูปแบบต่อไปนี้

structure-name . member

ตัวอย่าง เช่น เมื่อต้องการอ้างถึงสมาชิกภายใน struct ของ pt อยู่ตรงกับจุดใดบนแกนโคออดิเนตสามารถทำได้ดังนี้

printf (“%d, %d”, pt.x, pt.y);

ตัวอย่างโปรแกรมที่ 10.1 โปรแกรมเพื่อรับค่าจุดบนแกนโคออดิเนตของรูปสี่เหลี่ยม และนำมาทำการคำนวณหาพื้นที่ของรูปสี่เหลี่ยม

```
#include <stdio.h>
#include <conio.h>
typedef struct {
        int x;
        int y;
        } point;
void main() {
        point pt1,pt2 ;
        int area;
        clrscr();
```

ต่อจากด้านบน

```
/* Input Data */
printf("\nEnter rectangle data\n");
printf("\tStart point x : ");
scanf("%d", &pt1.x);
printf("\tStart point y : ");
scanf("%d", &pt1.y);
printf("\tEnd point x : ");
scanf("%d", &pt2.x);
printf("\tEnd point y : ");
scanf("%d", &pt2.y);
/* Calulate Rectangle */
area = (pt2.x-pt1.x) * (pt2.y-pt1.y);
printf("\nArea is %d", area);
getch();
}
```

ผลการทำงานโปรแกรม

ตัวอย่างโปรแกรมที่ 10.2 โปรแกรมเพื่อรับข้อมูลของผู้ใช้ซึ่งประกอบด้วย ชื่อ นามสกุล ความสูง และน้ำหนัก จากผู้ใช้โปรแกรม แล้วนำไปทำการคำนวณค่าดัชนีของน้ำหนัก (Body Mass Index : BMI) ซึ่งสามารถคิดได้จากสูตร BMI = w / h2 โดยที่ w แทนน้ำหนักตัวมีหน่วยเป็นกิโลกรัม และ h แทนความสูงมีหน่วยเป็นเมตร โดยผลการคำนวนค่า BMI อยู่ในช่วง 20-25 ให้โปรแกรมแสดงข้อความว่า “Normal BMI.” แต่ถ้าผลการคำนวณค่า BMI ยู่นอกช่วงดังกล่าวให้แสดงข้อความว่า “Dangerous BMI.” โดยกำหนดให้ใช้โครงสร้างทำการเก็บข้อมูลของผู้ใช้ ค่า BMI และผลลัพธ์ที่ได้ และให้แสดงข้อมูลทั้งหมด

```
#include <stdio.h>
#include<conio.h>
#include <string.h>
typedef struct {
        char name[16];
        char surname[20];
        float height;
        float weight;
        float bmi;
        char answer[14];
} student;
```

ต่อจากด้านบน

```
void main() {  clrscr();
        student std;
        printf("Enter student data\n");
        printf("\tName : ");
        scanf("%s", std.name);
        printf("\tSurname : ");
        scanf("%s", std.surname);
        printf("\tHeight (m) : ");
        scanf("%f", &std.height);
        printf("\tWeight (Kg) : ");
        scanf("%f", &std.weight);
        std.bmi = std.weight / (std.height * std.height);
        if (std.bmi >= 20 && std.bmi <= 25)
                strcpy(std.answer, "Normal BMI");
        else
                strcpy(std.answer, "Dangerous BMI");
        printf("\n\nBMI result");
        printf("\n%s %s weight %.2f kg. height %.2f", std.name,
std.surname, std.weight, std.height);
        printf("\n\tBody mass index %.2f is %s", std.bmi, std.answer);
        getch();
}
```

ผลการทำงานโปรแกรม

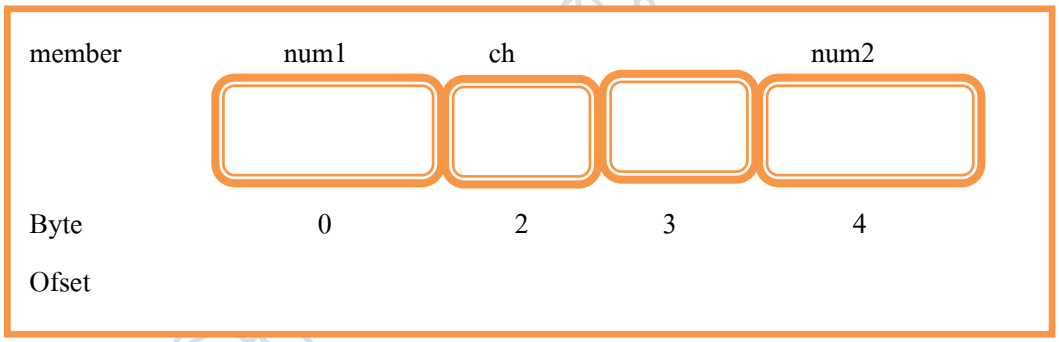
จากตัวโปรแกรม 10.2 ในการรับข้อมูลของข้อมูลสตริงด้วยคำสั่ง scanf นั้น เช่น ชื่อ (std.name) และนามสกุล (std.surname) ไม่ต้องใช้เครื่องหมาย & หน้าตัวแปรนั้น เนื่องจาก name และ surname เป็นตัว แปรอาร์เรย์ชนิดตัวอักขระ การอ้างถึงชื่อจะเป็นการอ้างถึงแอดเดรสของตัวแปรนั้น ซึ่งแตกต่างกับข้อมูลที่เป็นตัวแปรชนิดจำนวนเต็ม ต้องมีเครื่องหมาย & หน้าตัวแปรนั้น เช่น (std.weight) และความสูง (std.height) เป็นต้น

การเก็บข้อมูลแบบโครงสร้าง

การเก็บข้อมูลแบบโครงสร้าง การจัดการหน่วยความจำภายในจะทำการจัดเก็บข้อมูลตามลำดับที่มีการประกาศสมาชิกของข้อมูลนั้น โดยทั่วไปข้อมูลแบบโครงสร้างจะประกอบขึ้นจากข้อมูลหลาย ๆ ชนิด และข้อมูลแต่ละชนิดมักจะมีการจองพื้นที่ใช้งานที่แตกต่างกัน เนื่องด้วยจากการจองพื้นที่ของหน่วยความจำภายในระบบส่วนใหญ่ ทำการจองแอดเดรส (address) ตามขนาดที่เก็บของชนิดของตัวแปร เช่น ขนาด 2 หรือ 4 ไบท์ ดังตัวอย่าง

```
typedef struct {
        int num1;
        char ch;
        int num2;
} alignment;
alignment example; /*ประกาศตัวแปร example มีโครงสร้างแบบ alignment */
```

ข้อมูลประเภท int จะต้องใช้แอดเดรส ใช้พื้นที่ 2 ไบต์ในการเก็บข้อมูล ส่วนข้อมูลชนิด char จะใช้พื้นที่ใดก็ได้ การจองพื้นที่ในหน่วยความจำให้กับตัวแปร example รูปที่ 10.2



รูปที่ 10.2 แสดงการจองพื้นที่หน่วยความจำของตัวแปร example

จากรูปที่ 10.2 ตัวแปร num2 จะไม่สามารถใช้พื้นที่ที่ติดกับ ch ได้ เนื่องจาก num2 เป็นตัวแปรชนิดข้อมูลประเภทตัวเลขที่จะต้องใช้แอดเดรสขนาดในหน่วยความจำ 2 ไบท์ ดังนั้นการจองพื้นที่ของตัวแปร example จึงทำให้เกิดที่ว่างที่ไม่สามารถนำมาใช้ประโยชน์ได้ ดังนั้นการประกาศสมาชิกของโครงสร้างจะมีผลต่อการใช้พื้นที่ในหน่วยความจำของระบบ

ข้อมูลแบบโครงสร้างกับฟังก์ชัน

ตัวแปรโครงสร้างกับฟังก์ชัน ข้อมูลที่นำมาใช้งานร่วมกันสามารถทำได้หลายลักษณะ เช่นเดียวกับการใช้งานตัวแปรทั่วไป กล่าวคือ สามารถใช้ฟังก์ชันคืนค่าเป็น struct และการส่งอาร์กิวเมนท์ให้ฟังก์ชันเป็นตัวแปร โครงสร้างได้ ตัวอย่างที่ 10.3

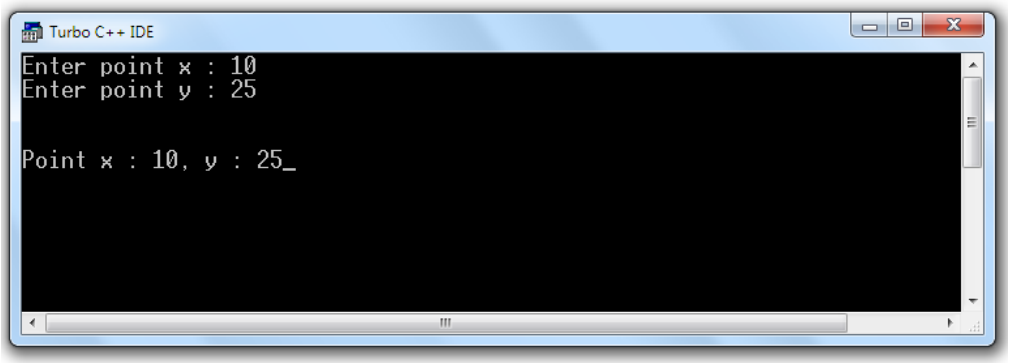
ตัวอย่างโปรแกรมที่ 10.3 การรับข้อมูลและแสดงผลข้อมูลตัวแปรโครงสร้างโดยใช้ฟังก์ชัน

```
  #include <stdio.h>
#include <conio.h>
struct point {
        int x;
        int y;
};
struct point readPoint( );
void printPoint(struct point);
void main() {
        struct point pt;
        pt = readPoint( );
        printPoint(pt);
        }
struct point readPoint( ) {
```

ต่อจากด้านบน

```
struct point p1;
        clrscr();
        printf("Enter point x : ");
        scanf("%d", &p1.x);
        printf("Enter point y : ");
        scanf("%d", &p1.y);
        return (p1);
}
void printPoint(struct point p) {
        printf("\n\nPoint x : %d, y : %d", p.x, p.y);
        getch();
}
```

ผลการทำงานโปรแกรม



การใช้พอยน์เตอร์กับตัวแปรโครงสร้าง

การใช้ฟังก์ชันให้มีการคืนค่าเป็นตัวแปรโครงสร้าง หรือการส่งอาร์กิวเมนท์เป็นตัวแปรโครงสร้างไปยัง ฟังก์ชันไม่เหมาะกับตัวแปรโครงสร้างที่มีขนาดใหญ่ เนื่องด้วยทุกครั้งที่ส่งตัวแปรโครงสร้างจะเป็นการจองพื้นที่ตัวแปรใหม่ขึ้นในฟังก์ชัน และมีการสำเนาค่าไปยังตัวแปรตัวใหม่ในฟังก์ชัน ซึ่งจะทำให้ช้าและเปลืองพื้นที่หน่วยความจำ ซึ่งสามารถแก้ปัญหาได้โดยใช้พอยน์เตอร์มาช่วยแก้ปัญหา โดยส่งแอดเดรสของตัวแปรโครงสร้างมายังฟังก์ชันซึ่งรับอาร์กิวเมนท์เป็นพอยน์เตอร์ อาร์กิวเมนท์จะชี้ไปยังแอดเดรสเริ่มต้นของตัวแปรโครงสร้างจะช่วยให้การทำงานเร็วขึ้นและไม่ต้องจองหน่วยความจำ กล่าวคือโครงสร้างนั้นมีขนาดเท่าใดก็จะใช้พื้นที่และเวลาในการกำหนดแอดเดรสเท่ากัน เสมอ แต่สิ่งที่ต้องระวังคือหากมีการเปลี่ยนแปลงค่าที่พอยน์เตอร์ชี้อยู่ ค่าในตัวแปรโครงสร้างที่ส่งมายังฟังก์ชันจะเปลี่ยนตามโดยอัตโนมัติ การประกาศตัวแปรพอยน์เตอร์ชี้ไปยัง struct หลักการดังนี้

```
struct point *pp;
```

ตัวแปรพอยน์เตอร์ pp ชี้ไปยังข้อมูลแบบโครงสร้างชื่อ struct point การเขียน *pp จะเป็นการอ้างถึงโครงสร้างการอ้างถึงสมาชิกสามารถทำได้โดยอ้าง (*pp).x หรือ (*pp).y ตัวอย่าง

```
struct point origin={30, 20};
struct point *pp;
pp = &original;
printf ( “origin is (%d, %d)\n”, (*pp).x, (*pp).y );
```

ข้อสังเกต คือ (*pp).x จะไม่เหมือนกับ *pp.x เนื่องจากคำสั่งเครื่องหมาย . (จุด) จะมีลำดับความสำคัญทางการคำนวณคณิตศาสตร์ที่สูงกว่าเครื่องหมาย * การการแปลความหมาย *pp.x จะเหมือนกับการอ้าง *(pp.x) ซึ่งอาจทำให้เกิดความผิดพลาดขึ้น การอ้างถึงสมาชิกในตัวแปรพอยน์เตอร์ชี้ไปยังโครงสร้างอาจเขียนอีกลักษณะหนึ่งโดยใช้เครื่องหมาย -> สมมติ p เป็นพอยน์เตอร์ รูปแบบการใช้เป็นดังนี้

```
p->member-of-structure
```

จะสามารถแปลงประโยคการใช้พอยน์เตอร์อ้างสมาชิกของโครงสร้างจากตัวอย่างข้างบนได้ว่า

```
printf ( “origin is (%d, %d)\n”, pp->x, pp->y);
```

การอ้างถึง (*pp).x จะมีผลเหมือนกับการอ้างถึง pp->x ดังนี้

```
struct rect r, *pr = r;
```

การอ้างถึงสมาชิกต่อไปนี้จะมีผลเท่ากับการอ้างถึงสมาชิกตัวเดียวกัน

```
r.pt1.x
(*pr).pt1.x
pr->pt1.x
```

ตัวอย่างโปรแกรมที่ 10.4 แสดงตัวอย่างการอ้างใช้ตัวแปรพอยน์เตอร์กับโครงสร้าง

```
#include <stdio.h>
#include <conio.h>
struct point {
int x;
int y;
};
struct rectangle {
struct point pt1;
struct point pt2;
};
void main() { clrscr();
        struct point origin={30,20};
        struct point *pp;
        struct rectangle r={{30,20},{100,200}};
        struct rectangle *pr;
        pp = &origin;
        printf("X : %d, Y : %d", (*pp).x, (*pp).y);
        pr = &r;
        printf("\nr.pt1.x = %d", r.pt1.x);
        printf("\n(*pr).pt1.x = %d", (*pr).pt1.x);
        printf("\npr->pt1.x = %d", pr->pt1.x);
        getch();
}
```

ผลการทำงานโปรแกรม

จากโปรแกรมที่ 10.4 เป็นโปรแกรมที่ทำการส่งใช้ตัวแปรพอยน์เตอร์ในรูปของฟังก์ชัน โดยการส่งแอดเดรสของตัวแปรโครงสร้างไปยังฟังก์ชันนั้น สามารถทำได้เหมือนกับการใช้ตัวแปรโดยทั่วไป คือ

```
struct point pt;
readPoint ( &pt );
```

ภายในฟังก์ชัน main( ) มีการเรียกใช้ฟังก์ชัน readPoint( ) โดยส่งแอดเดรสของตัวแปรโครงสร้าง pt ไปยังฟังก์ชัน โปรโตไทป์ของฟังก์ชัน readPoint จะต้องประกาศ

```
void readPoint ( struct point * ) ;
```

การประกาศฟังก์ชัน readPoint( ) จะทำโดยใช้คำสั่ง

```
void readPoint (struct point *pPt);
```

กระบวนการที่เกิดขึ้น คือ จะมีการจองพื้นที่หน่วยความจำให้กับตัวแปรพอยน์เตอร์ pPt และมีการสำเนาค่าแอดเดรสของตัวแปรโครงสร้าง pt ในฟังก์ชัน main( ) มาเก็บยัง pPt หรือกล่าวได้ว่าให้ตัวแปร pPt ชี้ไปยังตัวแปรโครงสร้าง pt นั่นเอง

ยูเนียน (Union)

ยูเนียนเป็นการจัดการข้อมูลเป็นชุดชนิดข้อมูลที่คล้ายกับชนิดข้อมูลโครงสร้าง คือ เป็นชนิดข้อมูลที่ประกอบด้วยสมาชิกที่มีชนิดข้อมูลหลาย ๆ ประเภท การใช้งานทุกต่าง ๆ จะใช้ในลักษณะเดียวกับข้อมูลโครงสร้าง ทั้งการอ้างถึงสมาชิก การส่งยูเนียนให้กับฟังก์ชันและอื่น ๆ แต่ต่างกันที่ในขณะใดขณะหนึ่งนั้นยูเนียนจะสามารถใช้สมาชิกได้เพียงตัวเดียวเท่านั้น ยูเนียนจะช่วยในการจัดการข้อมูลที่ต่างชนิดกันในพื้นที่ของตัวแปรยูเนียนเดียวกัน เช่น

```
union number { /*members are overlaid */
int integer;
float decimal;
};
typedef union number Number;
Number data;
```

การจองพื้นที่ของตัวแปร data จะจองให้กับสมาชิกของยูเนียนทั้ง 2 ตัว แต่ในขณะใดขณะหนึ่งจะสามารถใช้สมาชิกได้แค่ตัวเดียวเท่านั้น ซึ่งผู้เขียนโปรแกรมจะต้องเป็นผู้ควบคุมการใช้ด้วยตัวเอง คอมไพเลอร์จะไม่ตรวจสอบความผิดพลาดให้ แสดงตัวอย่างการใช้งาน ตัวอย่างที่ 10.5

ตัวอย่างโปรแกรมที่ 10.5 ตัวอย่างการใช้สมาชิกของยูเนียน

```
#include <stdio.h>
#include <conio.h>
union number {
int integer;
float decimal;
        };
typedef union number Number;
void main ( ) { clrscr();
        Number data;
        data.integer = 20000;
        printf ( "int : %6d, float : %10.4f\n", data.integer, data.decimal );
        data.decimal = 123.0;
        printf ( "int : %6d, float : %10.4f\n", data.integer, data.decimal );
        getch();
}
```

ผลการทำงานโปรแกรม

จากโปรแกรม 10.5 ผลการทำงานจะขึ้นอยู่กับการทำงานแต่ละครั้ง แต่การใช้ฟังก์ชัน printf ครั้งแรกจะให้คำตอบจำนวนเต็มที่ถูกต้องในขณะที่จำนวนจริงจะได้ค่าที่ไม่สามารถคาดเดาได้ และการใช้ฟังก์ชัน printf ครั้งที่ 2 จะให้คำตอบจำนวนจริงที่ถูกต้องในขณะที่จำนวนเต็มจะได้ค่าที่ไม่สามารถคาดเดาได้ เช่น

int : 20000, float : -0.0000
int : 0, float : 123.0000

การอ้างถึงสมาชิกภายในยูเนียนสามารถใช้

union-variable.member-name

หรือ

union-variable->member-name

ดังตัวอย่างการใช้งานตัวอย่างโปรแกรมที่ 10.6 โปรแกรมแสดงการใช้งานยูเนียน

```
#include <stdio.h>
#include <conio.h>
typedef enum { INT, FLOAT } Tag;
union number { /* number pair template */
int integer;
float decimal;
};
typedef union number Number;
struct pair { /* tagged number pairs */
        Tag tagged;
        Number value;
        };
typedef struct pair Pair;
void print_pair ( Pair );
void main ( ) { clrscr();
        Pair data;
        data.tagged = INT;
        data.value.integer = 20000;
        print_pair ( data );
        data.tagged = FLOAT;
        data.value.decimal = 123.0;
        print_pair ( data );
        getch();        }
```

ผลการทำงานโปรแกรม

จากตัวอย่างโปรแกรมที่ 10.6 สามารถกำหนดชนิดข้อมูลใหม่ขึ้นได้เอง ด้วยคำสั่ง

```
enum workday { Monday, Tuesday, Wednesday, Thursday, Friday };
```

เป็นการกำหนดชนิดข้อมูลชื่อ Workday ประเภทข้อมูลที่เป็นไปได้คือข้อมูลที่อยู่ในเครื่องหมายปีกกาทั้งหมด

เมื่อต้องการใช้งานต้องประกาศตัวแปรดังนี้

```
enum workday today;
today = Monday;
if (today == Monday)
printf("Today is %d", today);
```

ผลลัพธ์ที่ได้จากการทำงาน คือ Today is 0 เนื่องจากการเก็บข้อมูลของ enum จะเก็บในลักษณะคล้ายตัวแปรชุดของจำนวนเต็ม โดยที่ข้อมูลตัวแรกมีค่าเป็น 0 และไล่ค่าเป็น 1 2 3 ไปจนกระทั่งหมดข้อมูล และสามารถใช้ enum ร่วมกับคำสั่ง typedef ด้วย

สรุป

โครงสร้างข้อมูลมีประโยชน์อย่างมาก ในการจัดเก็บข้อมูลที่มีความแตกต่างกันของแต่ละชนิด (Type) ให้สามารถนำมาเก็บเป็นข้อมูลชุดเดียวกันหรือกลุ่มเดียวกันได้ กล่าวคือในหนึ่งโครงสร้างข้อมูลสามารถทำการเก็บข้อมูลที่เป็นชนิดตัวเลขจำนวนเต็ม ตัวเลขทศนิยม ตัวอักขระ หรือตัวอักษรได้ ซึ่งแตกต่างจากการเก็บข้อมูลด้วย อาร์เรย์ที่มีข้อจำกัดด้านชนิดของข้อมูล ต้องเป็นชนิดเดียวกันในการเก็บในหนึ่งอาร์เรย์เท่านั้น ดังนั้นโครงสร้างข้อมูล สามารถนำมาประยุกต์ใช้ในการจัดเก็บข้อมูลได้เป็นระบบและสามารถตอบสนองการเก็บอย่างเป็นครบถ้วนทุกชนิดตัวแปรมาเป็นโครงสร้างเดียวกัน

แบบฝึกหัดท้ายบทที่ 10

ตอนที่ 1 จงเติมคำหรือข้อความในช่องว่างต่อไปนี้ให้ถูกต้อง

1. คีย์เวิร์ดใดที่ใช้ในการสร้างโครงสร้างข้อมูลของภาษาซี

   .....

2. struc ของภาษาซีมี กี่ชนิดประกอบด้วยอะไรบ้าง

   .....

3. การอ้างถึงข้อมูลที่เป็นสมาชิกของโครงสร้าง ต้องใช้เครื่องหมายใด จงยกตัวอย่าง

   .....

4. จงเขียนโครงสร้างให้ประกอบด้วยตัวแปรเลขจำนวนเต็ม 1 ตัวแปร และตัวแปรเลขทศนิยม 1 ตัวแปร

   .....

5. จงประกาศโครงสร้างใหม่ 1 โครงสร้างโดยยึดโครงสร้างจากข้อ 4

   .....

6. จงให้ความหมาย struct point original = {10, 5}; คือ

   .....

7. จงให้ความหมาย struct point **pp; คือ

   .....

8. จงให้ความหมายจาก pp = &original; คือ

   .....

9. จงให้ความหมายจาก printf(“original is (%d, %d)\n”, (*pp).x (*pp).y); คือ

   .....

10. จงให้ความหมายจาก p->member-of-structre คือ

    .....

ตอนที่ 2 จงทำเครื่องหมายกากบาท (x) ทับหน้าข้อที่ถูกต้องที่สุด
โดยใช้โค๊ดด้านล่างนี้ตอบคำถาม

```
1  #include <stdio.h>
2  #include <conio.h>
3  struct point {int x;   int y;};
4  struct rectangle {struct point pt1; struct point pt2;};
5  void main() { clrscr();
6     struct point origin={30,20};
7     struct point *pp;
8     struct rectangle r={{30,20},{100,200}};
9     struct rectangle *pr;
10    pp = &origin;
11    printf("X : %d, Y : %d", (*pp).x, (*pp).y);
12    pr = &r;
13    printf("\nr.pt1.x = %d", r.pt1.x);
14    printf("\n(*pr).pt1.x = %d", (*pr).pt1.x);
15    printf("\npr->pt1.x = %d", pr->pt1.x);
16    getch(); }
```

1. บรรทัดใดเป็นการประกาศโครงสร้างที่มีอากูเมนต์เป็นตัวแปรตัวเลขจำนวนเต็ม 2 ตัวแปร

   ก. 3 ข. 4

   ค. 6 ง. 7

2. บรรทัดใดเป็นการประกาศโครงสร้างที่มีอากูเมนต์เป็นโครงสร้าง 2 โครงสร้าง

   ก. 3 ข. 4

   ค. 6 ง. 7

3. บรรทัดใดเป็นการสร้างโครงสร้างและกำหนดตัวแปรตัวเลขจำนวนเต็ม 2 ตัวแปร

ก. 3 ข. 4

ค. 6 ง. 7

4. บรรทัดใดเป็นการสร้างโครงสร้างโดยมีส่วนประกอบเป็นพอยน์เตอร์

ก. 3 ข. 4

ค. 6 ง. 7

5. บรรทัดใดเป็นการสร้างโครงสร้างและกำหนดตัวแปรตัวเลขจำนวนเต็ม 2 ตัวแปรอยู่ภายใต้โครงสร้างแต่ละโครงสร้าง จำนวน 2 โครงสร้าง

ก. 8 ข. 9

ค. 10 ง. 11

6. บรรทัดใดเป็นการสร้างพอยน์เตอร์เพื่อใช้ในการเก็บที่อยู่ของโครงสร้าง

ก. 8 ข. 9

ค. 10 ง. 11

7. บรรทัดใดเป็นนำที่อยู่ของโครงสร้าง มาเก็บที่พอยน์เตอร์

ก. 8 ข. 9

ค. 10 ง. 11

8. บรรทัดใดสั่งให้แสดงค่าตัวแปรที่อยู่ในโครงสร้าง มาแสดง

ก. 8 ข. 9

ค. 10 ง. 11

9. บรรทัดใดเป็นนำที่อยู่ของโครงสร้าง มาเก็บที่พอยน์เตอร์

ก. 12 ข. 13

ค. 14 ง. 15

10. บรรทัดใดสั่งให้แสดงค่าตัวแปรที่อยู่ในโครงสร้าง มาแสดง

ก. 12 ข. 13

ค. 14 ง. 15

ตอนที่ 3 จงทำการวิเคราะห์คำถามและทำการเขียนอภิปรายคำตอบตามที่ผู้อ่านเข้าใจโดยยึดความถูกต้องของเนื้อหาประกอบการบรรยาย

1. ให้ผู้อ่านเขียนการประกาศโครงสร้าง ให้ถูกต้อง

```
struct {

char address[20]

int num

}
```

2. ให้ผู้อ่านเขียนแก้ไขโครงสร้าง ให้ถูกต้อง

```
struct data {
int x;
int y;
};
data p1; . .
printf("\np1 = %d", p1);
```

3. ให้ผู้อ่านเขียนโครงสร้างเพื่อใช้ในการเก็บข้อมูลดังนี้ ไปนี้

   ยี่ห้อ ……..

   รุ่น ……..

   ประเภท ……

   ผลิตจากประเทศ ……

4. ให้ผู้อ่านเขียนเพื่อเก็บข้อมูลรถยนต์จำนวน 5 คัน โดยที่เก็บข้อมูลยี่ห้อ รุ่น ปีที่ผลิต ความเร็วสูงสุด โดยให้โปรแกรมทำงานดังนี้ 1) แสดงรายละเอียดของรถยนต์ที่จัดเก็บทั้งหมด 2) แสดงรถยนต์ที่มีความเร็วสูงที่สุด กรณีที่มีรถยนต์ที่มีความเร็วสูงสุดเท่ากัน ให้พิมพ์รายละเอียดของรถยนต์นั้นทั้งหมด 3) รับข้อมูลปีที่ผลิต จากผู้ใช้งานโปรแกรม เพื่อทำการค้นหาและแสดงผลข้อมูลของรถยนต์

5. ให้ผู้อ่านเขียนโปรแกรมที่สามารถแสดงผลเงินรายได้ของพนักงานทั้งหมด โดยแยกแสดงผลตามประเภทของพนักงานที่ใช้โครงสร้างและยูเนียน เพื่อเก็บข้อมูลเงินรายได้ของพนักงานไม่เกิน 50 คน หากมีการป้อนข้อมูลเงินเดือนของพนักงานน้อยกว่า 6,000 บาทจะเป็นการหยุดการป้อนข้อมูล องค์กรแห่งนี้แบ่งพนักงานเป็น 2 ประเภท คือ พนักงานทั่วไป และพนักงานฝ่ายขาย พนักงานทั่วไปมีรายได้จากเงินเดือน และค่าทำงานนอกเวลา (Over Time) พนักงานฝ่ายขายจะมีรายได้จากเงินเดือน และค่าคอมมิสชั่น

เอกสารอ้างอิง

ศรัณย์ อินทโกสุม (2539). ทฤษฎีและตัวอย่างโจทย์การเขียนโปรแกรมด้วยภาษาซี กรุงเทพฯ : แมคกรอฮิล อินเตอร์เนชั่นแนล เอ็นเตอร์ไพรส์, อิงค์.

ธันวา ศรีประโมง (2539). การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม. พิมพ์ครั้งที่ 4. กรุงเทพฯ : มหาวิทยาลัยเทคโนโลยีมหานคร.

วิจักษณ์ ศรีสัจจะเลิศวาจา และดุษฎี ประเสริฐธิติพงษ์ ออนไลน์ : www.satit.su.ac.th/soottin.

Brian, W. K. Programming in C: A Tutorial Online: http://www.lysator.liu.se/c/bwktutor.html.

Byron S. Gottfried, “Schaum ’s Theory and problems of programming with c” McGraw-Hill,Inc., 1990.

Kenneth A.Barclay. *“*ANSI C Problem Solving and Programming” Prentice Hall International Ltd., 1990.

Steven, H. & Lutfar, R. (2006). Art of Programming Contest: C Programming | Data Structure | Algorithms (ACM supported), 2nd Edition.