บทที่ 11 การสร้างชนิดของข้อมูล

(Construted Type)

ชนิดของโครงสร้างเป็นที่เก็บรวบรวมของตัวแปรที่มีความสัมพันธ์กัน ซึ่งอยู่ภายใต้ชื่อเดียวกัน โครงสร้างอาจจะประกอบด้วยตัวแปรที่มีข้อมูลต่างชนิดกัน ในทางตรงกันข้ามแถวลำดับประกอบด้วยสมาชิกที่มีข้อมูลชนิดเดียวกัน โดยทั่วไปโครงสร้างจะเป็นตัวกำหนดระเบียนที่เก็บไว้ในแฟ้มข้อมูล ตัวชี้และโครงสร้างอาจใช้เป็นส่วนประกอบพื้นฐานของโครงสร้างข้อมูลที่ซับซ้อน เช่น รายการโยง คิว สแตก และต้นไม้

11.1 คำจำกัดความของโครงสร้าง

โครงสร้าง เป็นชนิดข้อมูลที่สร้างขึ้นโดยใช้ข้อมูลชนิดอื่น ๆ เช่น

```
struct card {
        char *face;
        char *suit;
        };
```

คำสำคัญ struct ใช้เพื่อประกาศตัวระบุที่เป็นชนิดโครงสร้างแล้วตามด้วยป้ายระบุโครงสร้าง ในที่นี้ป้ายระบุโครงสร้างคือ card และ card เป็นชนิดข้อมูลโครงสร้าง ซึ่งข้อมูลชนิดโครงสร้าง card จะประกอบด้วยสมาชิกที่ประกาศอยู่ภายในเครื่องหมายวงเล็บปีกกา โดยสมาชิกของโครงสร้างประเภทเดียวกันต้องมีชื่อที่เป็นเอกลักษณ์ และคำจำกัดความของโครงสร้างจะต้องจบด้วยเครื่องหมาย “;”

จากตัวอย่าง card เป็นประเภทข้อมูลชนิดโครงสร้างประกอบด้วย สมาชิก 2 ตัว คือ *face และ *suit โดยทั่วไปสมาชิกของ struct นี้จะเป็นชนิดใดก็ได้ เช่น int float หรือ ตัวชี้ที่ชี้ไปยังแถวลำดับ ตัวชี้ที่ชี้ไปยังจำนวนเต็มและ ตัวชี้ที่ชี้ไปยังจำนวนจริง

การประกาศตัวแปรชนิดโครงสร้างสามารถทำได้โดยวิธีเดียวกันกับการประกาศตัวแปรชนิดอื่น

เช่น struct card a, deck [52], *cPtr;

คำสั่งนี้ประกาศให้ a เป็นตัวแปรชนิด struct card และ deck เป็นแถวลำดับมีสมาชิกเป็นชนิด struct card จำนวน 52 ตัว และ cPtr เป็นตัวชี้ที่ชี้ไปยัง struct card ในการประกาศตัวแปร struct card นี้ ในการประกาศตัวแปร a, deck และ *cPtr สามารถนำการประกาศรวมกันได้ดังนี้

```
struct card {
        char *face;
        char *suit;
        } a, deck [52], *cPtr;
```

การกำหนดค่าเริ่มต้นให้กับตัวแปรชนิดโครงสร้าง สามารถทำได้เช่นเดียวกับการกำหนดค่าให้แถวลำดับ เช่น

```
struct card a= {“three“, “hearts”};
```

คำสั่งนี้สร้างตัวแปร a ให้มีชนิด struct card ตัวแปร a นั้นมีสมาชิก 2 ตัวคือ face และ suit เรากำหนดค่าเริ่มต้นให้สมาชิก face มีค่าเป็น “three” และ suit มีค่าเป็น “hearts” ในกรณีที่ค่าที่กำหนดมีจำนวนน้อยกว่าจำนวนสมาชิก สมาชิกที่เหลือจะมีค่าเป็นศูนย์โดยอัตโนมัติ หรือเป็น null ถ้าเป็นตัวชี้

11.2 การเข้าถึงสมาชิกของข้อมูลชนิดโครงสร้าง

การเข้าถึงสมาชิกของข้อมูลชนิดโครงสร้างอาจใช้ตัวดำเนินการ 2 ตัวคือ ตัวดำเนินการจุด “.” หรือตัวดำเนินการตัวชี้ของโครงสร้าง “ -> ” ที่เรียกว่าตัวดำเนินการลูกศร การเข้าถึงสมาชิกของข้อมูลชนิดโครงสร้างทำได้โดยเรียกชื่อตัวแปรชนิดโครงสร้าง เช่น ต้องการพิมพ์ค่าภายในของ suit ใน structure a ที่ได้กล่าวมาแล้ว สามารถทำได้โดยใช้คำสั่ง

```
printf (“%s”, a.suit);
```

นิพจน์ *aPtr->suit นั้นมีความหมายเหมือนกันกับ (*aPtr).suit ซึ่งอ้างถึงค่าภายในของที่อยู่และเข้าถึงสมาชิก suit กรณีนี้ใช้เครื่องหมายวงเล็บเพราะตัวดำเนินการจุด (.) มีลำดับการทำงานสูงกว่าตัวดำเนินการตัวชี้ “*”

ตัวอย่างที่ 11.1 โปรแกรมแสดงการดำเนินงานของสมาชิกและตัวชี้ของข้อมูลชนิดโครงสร้าง โดยกำหนดมูลค่า “ace” และ “spades” ให้สมาชิกของข้อมูลชนิดโครงสร้าง a ตามลำดับ สำหรับตัวชี้ aPtr ได้รับค่าที่อยู่ของโครงสร้าง a คำสั่ง printf ทำหน้าที่พิมพ์สมาชิกของตัวแปร a

```
/* Using the structure member and structure pointer operators */
#include <stdio.h>
struct card {
        char *face;
        char *suit;
};
main()
{
        struct card a;
        struct card *aPtr;
        a.face = "Ace";
        a.suit = "Spades";
        aPtr = &a;
        printf("%s %s %s\n%s %s %s\n%s %s %s\n",
                a.face, " of ", a.suit,
                aPtr -> face, " of ", aPtr -> suit,
                (*aPtr).face, " of ", (*aPtr).suit);
        return 0;
} /* End main */
```

ผลลัพธ์ที่ได้

```
Ace  of  Spades
Ace  of  Spades
Ace  of  Spades
```

11.3 คำสำคัญ typedef

คำสำคัญ typedef ใช้ในการสร้างหรือกำหนดชนิดข้อมูล การประกาศชนิดข้อมูลโครงสร้างกำหนดโดย typedef จะทำให้ลดขั้นตอนในการประกาศตัวแปร

```
typedef struct card Card;
```

คำสั่งนี้กำหนดให้ Card เป็นข้อมูลชนิดใหม่ซึ่งเหมือนกับชนิด struc card ดังนั้น ป้ายระบุโครงสร้างจึงไม่จำเป็นต้องใช้ในที่นี้ เช่น

```
typedef struct {
        char *face;
        char *suit;
        } Card;
```

คำสั่งที่กล่าวไปนี้สร้างชนิดข้อมูลแบบโครงสร้างชื่อ Card ดังนั้น เราสามารถใช้ Card ประกาศตัวแปรที่มีชนิดเป็น struct card เช่น

Card deck [52];

คำสั่งนี้ประกาศแถวลำดับของชนิดข้อมูลโครงสร้างมีสมาชิก 52 ตัว

11.4 ชนิดข้อมูล Union

Union เป็นชนิดข้อมูลที่สมาชิกของ Union แต่ละตัวจะถูกเก็บในตำแหน่งเดียวกัน สมาชิกของ Union จะเป็นข้อมูลชนิดใดก็ได้ แต่สมาชิกของ Union มักจะมีขนาดเล็ก ซึ่งทำให้ Union สามารถเก็บสมาชิกได้เป็นจำนวนมาก โดยทั่วไป Union มักจะประกอบด้วยข้อมูลมากกว่า 2 ชนิดขึ้นไป การเรียกใช้สมาชิกของ Union แต่ละครั้งทำได้ครั้งละ 1 ตัว และเพียง 1 ชนิดข้อมูลเท่านั้น นักเขียนโปรแกรมควรระวังการอ้างถึงข้อมูลใน Union นั้น เพื่อจะได้ใช้ชนิดข้อมูลได้ถูกต้องตามต้องการ

การประกาศ Union มีรูปแบบเช่นเดียวกับ structure ดังนี้

```
Union number {
        int x;
        float y;
        }
```

คำสั่งนี้แสดงว่า number มีชนิดเป็น Union ซึ่งประกอบด้วยสมาชิก คือ int x และ float y การประกาศ Union ในโปรแกรมจะอยู่ในตำแหน่งก่อนหน้า การประกาศ main() เสมอ

เราสามารถคัดลอกค่าภายในของ Union หนึ่งไปให้อีก Union หนึ่งได้โดยใช้เลขที่อยู่ (&) ของ Union และสามารถเข้าถึงสมาชิกของ Union โดยใช้ตัวดำเนินจุดของโครงสร้างและตัวดำเนินการตัวชี้ของโครงสร้าง แต่เราไม่สามารถเปรียบเทียบหนึ่ง Union กับ Union อื่นได้

การกำหนดค่าเริ่มต้นให้ Union สามารถทำได้โดยกำหนดมูลค่าที่มีชนิดเป็นชนิดเดียวกับสมาชิกตัวแรกของ Union เช่น

Union number valve = {10}

11.5 ตัวคงที่ Enumeration

ในภาษา C ผู้ใช้สามารถกำหนดชนิดข้อมูลได้เองซึ่งเราเรียกชนิดข้อมูลนี้ว่า enumeration การกำหนดข้อมูลชนิดนี้ใช้คำสำคัญ enum ซึ่งเป็นชนิดของข้อมูลของค่าคงตัว(enumeration constant)ของจำนวนเต็มโดยมูลค่าของ enum เริ่มต้นด้วย 0 และเพิ่มค่าครั้งละ 1 เช่น

enum months {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC};

คำสั่งนี้สร้างชนิดข้อมูลใหม่คือ enum months ซึ่งประกอบด้วยตัวระบุ 12 ตัว ตัวระบุจะถูกกำหนดให้เป็นจำนวนเต็มที่มีค่าตั้งแต่ 0 ถึง 11 โดยอัตโนมัติ แต่ถ้าต้องการกำหนดให้เดือนมีค่าจาก 1 ถึง 12 กำหนดดังนี้

```
enum months {JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP,
             OCT, NOV, DEC};
```

เนื่องจากกำหนดให้สมาชิกตัวแรกใน enum months มีค่าเป็น 1 ค่าที่เหลือจึงมีการเพิ่ม ค่าเป็น 2 ถึง 12 ตามลำดับ การกำหนดค่าให้ตัวระบุแต่ละตัวสามารถทำได้ดังโปรแกรมในตัวอย่างที่ 11.2 โปรแกรมนี้แสดงว่า ตัวแปร month ใช้คำสั่ง for เพื่อพิมพ์ชื่อเดือนจากแถวลำดับ monthName

ตัวอย่างที่ 11.2 โปรแกรมแสดงการพิมพ์ชื่อเดือนโดยใช้ชนิดข้อมูล enum

```
/* Using an enumeration type */
#include <stdio.h>
enum months {JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG,
             OCT, NOV, DEC};
main()
{
        enum months month;
        char *monthName[] = {"", "January", "February", "March"
                             "May", "June", "July", "August",
                             "September", "October", "Novem"
                             "December"};
        for (month = JAN; month <= DEC; month++)
                printf("%2d%11s\n", month, monthName[month]);
        return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
 1    January
 2   February
 3   MarchMay
 4       June
 5       July
 6     August
 7  September
 8    October
 9      Novem
10   December
11     (null)
```

ตัวดำเนินการบิตไวส์

ตัวดำเนินการบิตไวส์ดังได้กล่าวไปบ้างแล้วในบทที่ 3 สำหรับในบทนี้จะแสดงการทำงานของตัวดำเนินการบิตไวส์ เช่น

bitwise AND (&)

bitwise inclusive OR (|)

bitwise exclusive OR (^)

left shift (<<)

right shift (>>)

คอมพลีเมนต์ (~)

ตัวอย่างที่ 11.3 โปรแกรมพิมพ์สายอักขระบิตของจำนวนเต็มที่ไม่มีเครื่องหมาย (unsigned integer)

```
/* Printing an unsigned integer in bits */
#include <stdio.h>
main()
{
        unsigned x;
        void displayBits (unsigned);
        printf ("Enter an unsigned integer: ");
        scanf ("%u", &x);
        displayBits (x);
        return 0;
}/* End main */
void displayBits (unsigned value)
{
        unsigned c, displayMask = 1 << 15;
        printf ("%7u = ", value);
        for (c = 1; c<= 16; c++) {
                putchar (value & displayMask ? '1' : '0');
                value <<= 1;
                if (c % 8 == 0)
                        putchar (' ');
        } /* End for */
        putchar ('\n');
}/* End void displayBits */
```

ผลลัพธ์ที่ได้

```
Enter an unsigned integer: 65000
  65000 = 11111101 11101000
```

ฟังก์ชัน displayBits ใช้ตัวดำเนินการ bitwise AND เพื่อรวมตัวแปร value กับตัวแปร displayMask เข้าด้วยกันสำหรับตัวแปร displayMask ได้รับการกำหนดให้มีค่า 1<<15 เลื่อน 1 ไป 15 ตำแหน่ง(10000000 00000000) ตัวดำเนินการ left shift เลื่อนมูลค่า 1 จากตำแหน่งบิตขวาสุด ไปยังตำแหน่งที่มีค่าสูงกว่า คือบิตซ้ายสุด แล้วใส่บิต 0 ด้านขวาทุกบิตที่เหลือ

สำหรับข้อความสั่ง putchar (value & displayMask ? '1': '0'); ใช้ตรวจดูว่าจะพิมพ์ 1 หรือ 0 ที่บิตซ้ายสุดของตัวแปร value สมมติว่าตัวแปร value มีค่าเป็น 65000 (11111101 11101000) เมื่อ value และ displayMask รวมกันโดยใช้ตัวดำเนินการ & บิตทุกบิตยกเว้นบิตที่มีอันดับสูง(บิตซ้ายสุด) ในตัวแปร value จะ "masked off" (ถูกซ่อน) เพราะบิตทุกบิตที่นำไป AND กับ 0 จะได้ผลลัพธ์เป็น 0 ถ้าบิตซ้ายมีค่าเป็น 1 value & displayMask ให้พิมพ์ผลลัพธ์เป็น 1 ในกรณีอื่นจะพิมพ์ผลลัพธ์เป็น 0

ต่อมาตัวแปร value มีการเลื่อนบิตไปทางซ้าย 1 บิต โดยใช้นิพจน์ value <<= 1 (value = value << 1) ขั้นตอนเหล่านั้นมีการดำเนินการซ้ำกับแต่ละบิตในตัวแปร value

โปรแกรมในตัวอย่างที่ 11.4 แสดงการใช้ตัวดำเนินการ bitwise AND, bitwise inclusive OR, bitwise exclusive OR และคอมพลีเมนต์ โปรแกรมใช้ฟังก์ชัน displayBits เพื่อพิมพ์มูลค่าของจำนวนเต็ม unsigned ผลลัพธ์ที่ได้จากตัวอย่างที่ 11.2

ตัวอย่างที่ 11.4 โปรแกรมแสดงการใช้ตัวดำเนินการ bitwise AND, inclusive OR, exclusive OR และคอมพลีเมนต์

```
/* Using the bitwise AND, bitwise inclusive OR, bitwise
   exclusive OR, and bitwise complement operators */
#include <stdio.h>
void displayBits (unsigned);
main()
{
        unsigned number1, number2, mask, setBits;
        number1 = 65535;
        mask = 1;
        printf ("The result of combining the following\n");
        displayBits (number1);
        displayBits (mask);
        printf ("using the bitwise AND operator & is\n");
        displayBits (number1 & mask);
        number1 = 15;
        setBits = 241;
        printf ("\nThe result of combining the following\n");
        displayBits (number1);
        displayBits (setBits);
        printf ("using the bitwise inclusive OR operator | is\n");
        displayBits (number1 | setBits);
        number1 = 139;
        number2 = 199;
        printf ("\nThe result of combining the following\n");
        displayBits (number1);
        displayBits (number2);
        printf ("using the bitwise exclusive OR operator ^ is\n");
        displayBits (number1 ^ number2);
        number1 = 21845;
        printf ("\nThe one's complement of \n");
        displayBits (number1);
        printf ("is\n");
        displayBits (~number1);
        return 0;
} /* end main */
 void displayBits (unsigned value)
{
        unsigned c, displayMask = 1 << 15;
        printf ("%7u = ", value);
        for (c = 1; c <= 16; c++)   {
                putchar (value & displayMask ? '1' : '0');
                value <<= 1;
                if (c % 8 == 0)
                        putchar ('  ');
        } /* End for */
        putchar (' \n');
} /* End void displayBits */
```

ผลลัพธ์ที่ได้

```
The result of combining the following
  65535 = 11111111 11111111
      1 = 00000000 00000001
using the bitwise AND operator & is
      1 = 00000000 00000001

The result of combining the following
     15 = 00000000 00001111
    241 = 00000000 11110001
using the bitwise inclusive OR operator | is
    255 = 00000000 11111111

The result of combining the following
    139 = 00000000 10001011
    199 = 00000000 11000111
using the bitwise exclusive OR operator ^ is
     76 = 00000000 01001100

The one's complement of
  21845 = 01010101 01010101
is
  43690 = 10101010 10101010
```

จากผลลัพธ์ที่ได้จากตัวอย่างที่ 11.4 ตัวแปรจำนวนเต็ม mask มีค่าเป็น 1 (00000000 00000001) และตัวแปร number1 กำหนดให้มีค่าเป็น 65535 (11111111 11111111) เมื่อนำ mask และ number1 มารวมกันโดยใช้ตัวดำเนินการ bitwise AND (&) โดยใช้นิพจน์ mask and number1 ผลลัพธ์ที่ได้คือ 00000000 00000001 บิตทุกบิตยกเว้นบิตที่มีลำดับต่ำในตัวแปร number1 ได้ถูกซ่อน (masked off) โดยการนำไป AND (&) กับตัวแปร mask

ในลำดับต่อมามีการดำเนินงานของตัวดำเนินการ inclusive OR ระหว่างตัวแปร number1 และ setBits การดำเนินงานของ exclusive OR ระหว่างตัวแปร number1 และ number2 และการดำเนินงานส่วนเติมเต็ม 1ของตัวแปร number1

ตัวอย่างที่ 11.5 โปรแกรมแสดงการทำงานของตัวดำเนินการ left shift (<<) และ right shift (>>)

```
/* Using the bitwise shift operators */
#include <stdio.h>
void displayBits (unsigned);
main()
{
        unsigned number1 = 960;
        printf ("\nThe result of left shifting\n");
        displayBits (number1);
        printf ("8 bit positions using the ");
        printf ("left shift operator << is\n");
        displayBits (number1 << 8);
        printf ("\nThe result of right shifting\n");
        displayBits (number1);
        printf ("8 bit positions using the ");
        printf ("right shift operator >> is\n");
        displayBits (number1 >> 8);
        return 0;
} /* end main */
void displayBits (unsigned value)
{
        unsigned c, displayMask = 1 << 15;
        printf ("%7u = ", value);
        for (c = 1; c <= 16; c++)  {
                putchar (value & displayMask ? '1' : '0');
                value <<= 1;
                if (c % 8 == 0)
                        putchar (' ');
        } /* End for */
        putchar ('\n');
} /* End void displayBits */
```

ผลลัพธ์ที่ได้

```
The result of left shifting
    960 = 00000011 11000000
8 bit positions using the left shift operator << is
  49152 = 11000000 00000000

The result of right shifting
    960 = 00000011 11000000
8 bit positions using the right shift operator >> is
      3 = 00000000 00000011
```

ฟังก์ชัน displayBits ใช้ในการพิมพ์มูลค่าของจำนวนเต็มชนิด unsigned ตัวดำเนินการ left shift (<<) เลื่อนบิตของตัวถูกดำเนินการที่อยู่ทางขวาไปทางซ้ายที่ละบิตตามจำนวนที่กำหนดไว้ทางด้านซ้าย สำหรับบิตด้านขวาที่ว่างเขียนแทนด้วย 0 ในกรณีที่บิต 1 เลื่อนไปจนเกินบิตซ้ายสุดนั้นจะตัดทิ้งไป ตัวอย่างที่ 11.5 ตัวแปร number1 ได้รับค่า 960 (00000011 11000000) ผลจากตัวดำเนินการ left shift กับตัวแปร number1 ขนาด 8 บิต ซึ่งเขียนโดยใช้นิพจน์ number 1 << 8 คือ 49152 (11000000 00000000)

ตัวดำเนินการ right shift จะเลื่อนบิตของตัวถูกดำเนินการที่อยู่ทางซ้ายไปทางขวาทีละบิตตามจำนวนที่กำหนดไว้ทางด้านขวา การทำงานนี้ทำให้มีการแทนที่บิตที่ว่างด้วย 0 และบิต 1 ที่เลื่อนไปทางขวาจนเกินบิตขวาสุดจะถูตัดทิ้งไป

แบบฝึกหัด

1. จงพิจารณาว่าข้อความดังต่อไปนี้ถูกหรือผิด ถ้าผิดจงอธิบาย
   ก. Structure ประกอบด้วยข้อมูลเพียงชนิดเดียวเท่านั้น
   ข. สามารถเปรียบเทียบ Union กับ Union ได้ว่าเท่ากันหรือไม่
   ค. การเขียนชื่อป้ายระบุของ Structure นั้นจะมีหรือไม่มีก็ได้
   ง. สมาชิกของโครงสร้างเดียวกันจะมีชื่อที่เป็นเอกลักษณ์
   จ. คำสำคัญ typedef ใช้กำหนดข้อมูลชนิดใหม่
   ฉ. ไม่สามารถเปรียบเทียบ structure กับ Structure ได้
2. จงเขียนคำสั่ง หรือชุดคำสั่งที่สามารถดำเนินงานได้ดังต่อไปนี้
   ก. กำหนดโครงสร้างที่เรียกว่า part ประกอบด้วย ตัวแปร partNumber ที่มีชนิดเป็น int และแถวลำดับ partName ชนิดเป็น char ซึ่งมีความยาวมากสุด 25 ตัวอักษร
   ข. กำหนด Part เป็น synonym สำหรับชนิด struct part
   ค. ใช้ Part ประกาศตัวแปร a เป็นชนิด struct part แถวลำดับ b[10] เป็นชนิด struct part และตัวแปร ptr เป็นชนิดตัวชี้ชี้ไปที่ struct part
   ง. อ่านข้อมูลนำเข้าเป็นหมายเลขของ part (part number) และชื่อ (part name) จากแผงแป้นอักขระไปเก็บไว้ในตัวแปร a
   จ. กำหนดค่าให้สมาชิกตัวที่ 3 ของ b มีค่าเหมือนกับตัวแปร a
   ฉ. กำหนดให้ที่อยู่ของแถวลำดับ b ชี้ไปที่ตัวแปร ptr
   ช. พิมพ์มูลค่าของสมาชิกตัวที่ 3 ของ แถวลำดับ b โดยใช้ตัวแปร ptr และตัวดำเนินการตัวชี้ที่อ้างถึงสมาชิก
3. จงเขียนโปรแกรมที่สามารถดำเนินการกับสมุดโทรศัพท์ส่วนบุคคล เมื่อป้อนข้อมูลนำเข้าที่ประกอบด้วย ชื่อ ที่อยู่ (บ้านเลขที่ ถนน ตำบล อำเภอ จังหวัด รหัสไปรษณีย์) และหมายเลขโทรศัพท์ ผู้ใช้สามารถเรียกดูหมายเลขโทรศัพท์ของผู้ใช้ทั้งหมดเมื่อกำหนดชื่อจังหวัด ชื่ออำเภอ รหัสไปรษณีย์ หรือชื่อคน และสั่งให้พิมพ์ข้อมูลทั้งหมด

4. จงพิจารณาว่าโปรแกรมนี้ทำหน้าที่อะไร

```
#include <stdio.h>
int mystery (unsigned);
main()
{
        unsigned x;
        printf ("Enter an integer : ");
        scanf ("%u", &x);
        printf ("The result is %d \n", mystery (x));
        return 0;
}
int mystery (unsigned bits)
{
        unsigned i, mask = 1 << 15, total = 0;
                for (i = 1; i <= 16; i++, bits <<= 1)
                        if ((bits & mask) == mask)
                                ++total;
                return total % 2 == 0? 1 : 0;
}
```

5. จงเขียนโปรแกรมที่แสดงบิตจากหลังมาหน้า โปรแกรมจะอ่าน input ที่ผู้ใช้กำหนดให้ แล้วเรียกใช้ฟังก์ชัน reverseBits เพื่อที่จะพิมพ์ข้อมูลส่งออกจากบิตหลังมาหน้า และพิมพ์ input