การใช้ pointer ในภาษาซีภายใต้ปริบทต่าง ๆ

1) ในปริบทของ array

ในการส่ง array ชนิด 1-D เป็น parameter นั้น สามารถประกาศได้ 3 แบบ คือ

```
int     main(void)
{       int     score[30], count;
        …
        sub1(score, &count);
}
void    sub1(int  value[30], int *n)      หรือ sub1(int  value[], int *n)
{                                         หรือ sub1(int  *value, int *n)
}
```

สังเกตุว่าในแบบสุดท้าย (ใช้ * ทั้งหมด) จะบอกไม่ได้ว่า ตัวใดเป็น array จริงและตัวใดเป็นเพียงตัวแปรโดด เพราะทั้งคู่เป็น address เหมือนกัน โดย pointer ที่แทน array จะเก็บ address ของสมาชิกตัวแรกของ array แต่ pointer ที่แทนตัวแปรเป็นเพียง address ของตัวแปรนั้น ๆ หากเขียนสลับกันเป็น sub1(&count, score) ในการเรียกฟังชัน sub1 จาก main แล้ว แม้ว่าจะ compile ผ่าน แต่ทำให้เกิดข้อผิดพลาดในระหว่างการ execute โปรแกรม เพราะหากเราต้องการเรียก element ตัวที่ 8 ของ value โดยใช้วิธีการคำนวณ offset (ดังที่ได้กล่าวไปแล้ว) ก็จะเกิด fault ขึ้นทันทีเพราะ value เป็น pointer (หรือ address ของ) ซึ่งชี้ไปที่ตัวแปรโดด (คือ count ใน main) ไม่มีสมาชิก เพราะไม่ใช่ array

2) ในปริบทของ structure (และ array ของ structure)

```
struct      tag
{
      int           data;
      struct   tag  *next;
};
```


```
typedef  struct  tag  RECORD;  /* RECORD is of type “struct tag” */
typedef  RECORD       *LINK;   /* LINK is of type “struct tag *” */
```

กรณีเดียวที่ไม่ต้องใส่ tag ใน structure ข้างต้น คือ ต้องประกาศตัวแปรชนิด structure ดังกล่าว พร้อมกับนิยามโครงสร้าง (structure template) เช่น

```
struct
{
      int           data;
      struct   tag  *next;
}  node;
```

ซึ่งไม่แนะนำให้ยึดถือเป็นแนวปฏิบัติ เพราะนิยามโครงสร้างดังกล่าวไม่สามารถนำไปใช้กับตัวแปรอื่นได้ ควรประกาศนิยามโครงสร้างก่อนดังตัวอย่างแรก แล้วจึงประกาศตัวแปร ซึ่งอาจจะใช้หรือไม่ใช้ typedef ช่วยเลยก็ได้

<u>วิธีการอ้างอิง</u>

จากตัวอย่างของ structure ข้างต้น การสร้างตัวแปรทำได้ 2 วิธีคือ

```
struct   tag     node;          /* variable ชนิด "struct tag"          */
struct   tag     *head, *temp;  /* pointer variable ชนิด "struct tag *" */
```

หรือใช้ชื่อใหม่ที่ตั้งขึ้นด้วยคำสั่ง typedef ข้างต้น

```
RECORD       node;          /* variable ชนิด "struct tag"                          */
LINK         head, temp;    /* pointer variable ชนิด "struct tag *" (ไม่มี * นำหน้า) */
```

<u>วิธีเขียน</u>

มีรูปแบบดังนี้

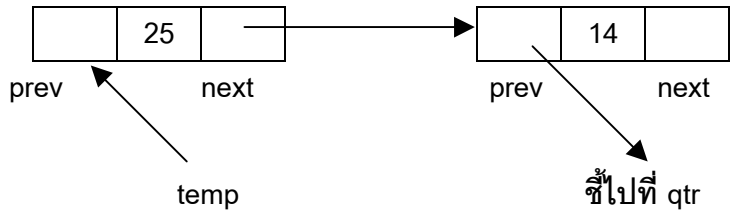
```
temp->data   = 3;       /* set "data" member = 3       */
temp->next   = NULL;    /* set "next" member = NULL    */
```

ในกรณีของ linked list ที่มีการอ้างอิงต่อเนื่อง ก็อาจจะดูซับซ้อนบ้าง ดังตัวอย่าง

```
temp->next->prev        = qtr;
```

ให้เริ่มพิจารณาจากซ้ายไปขวา โดยมอง temp->next เป็นเสมือน pointer 1 ตัว คือ

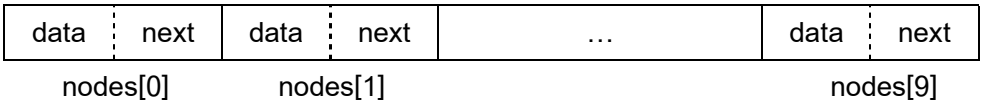
```
ptr->prev               = qtr;
```



กล่าวคือ ptr (temp->next) เป็น address ของ node 14 ซึ่งกำลังอ้างอิงค่าในช่อง (หรือ pointer ชื่อ) prev โดยกำหนดให้ชี้ไปยัง node ใหม่ที่มี address เป็น qtr ด้วยคำสั่ง assignment ข้างต้น

สำหรับ array of structure นั้น ก็ให้คิดในลักษณะเดียวกันกับ array ของ int, char, double โดยถือเสมือนว่า structure แต่ละอันเป็น element หนึ่งใน array เท่านั้น ดังตัวอย่าง

RECORD nodes[10];



การอ้างอิงแต่ละสมาชิกก็ทำในลักษณะเดียวกัน คือ nodes[8].data = 22 เป็นการกำหนด sub-member ชื่อ data ของสมาชิกลำดับที่ 9 ใน array nodes ให้มีค่าเป็น 22 หรือ nodes[4].next = qtr เป็นการกำหนด sub-member ซึ่งเป็น pointer ชื่อ next ของสมาชิกลำดับที่ 5 ใน array nodes ให้ชี้ไปที่ node อื่นที่ถูกชี้โดย qtr เป็นต้น

<u>การประยุกต์</u>

Structure มีประโยชน์ใช้งานในภาษาซีมาก เพราะสามารถรวม data type หลายๆ ชนิดเข้าด้วยกัน เป็นข้อมูล “ก้อน” เดียวกัน ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1

```
const int   Max_size    = 100;
enum        status
{
      EMPTY = -1,
      FULL  = Max_size – 1
};

struct      st
{
      int   data[Max_size];
      int   TOS;
};
struct      st    STACK;
. . .
STACK.TOS   = (int)EMPTY;          /* initialization of TOS      */
```

ตัวอย่างที่ 2

```
struct      dlink
{
      int               data;
      struct   dlink    *prev;
      struct   dlink    *next;
};
typedef     struct   dlink    DNODE;
typedef     struct   dlink    *DLINK;

DNODE       temp_swap;
DLINK       head = NULL, temp_ptr;
```

ตัวอย่างที่ 3

```
#define     Max_no      100
enum        que
{
      ENQUEUE     = 1,
      DEQUEUE     = 0
};

struct      qq
{
      int         data[Max_no];
      enum  que   queue_state;
};
struct      qq    FIFO_Q;
```

ตัวอย่างที่ 4

```
#define     Max         10
#define     Cap         50
struct      part
{
      int   data;                     /* critical section        */
      int   queue[Max];               /* queue size              */
      int   cond_var;                 /* to wait on this queue   */
};

struct      monitor
{
   int      local;                     /* local monitor data      */
   int      trigger_cond;              /* monitor condition var   */
   struct   part   channel[Cap];       /* 50 regular channels     */
   struct   part   urgent;             /* one urgent channel      */
};
struct      monitor     MONITOR;
```

นอกจากนี้ structure ยังใช้ในงานประยุกต์ระดับ bit ในฮาร์ดแวร์ รายละเอียดสามารถค้นคว้าได้จากหนังสือภาษาซีมาตรฐาน