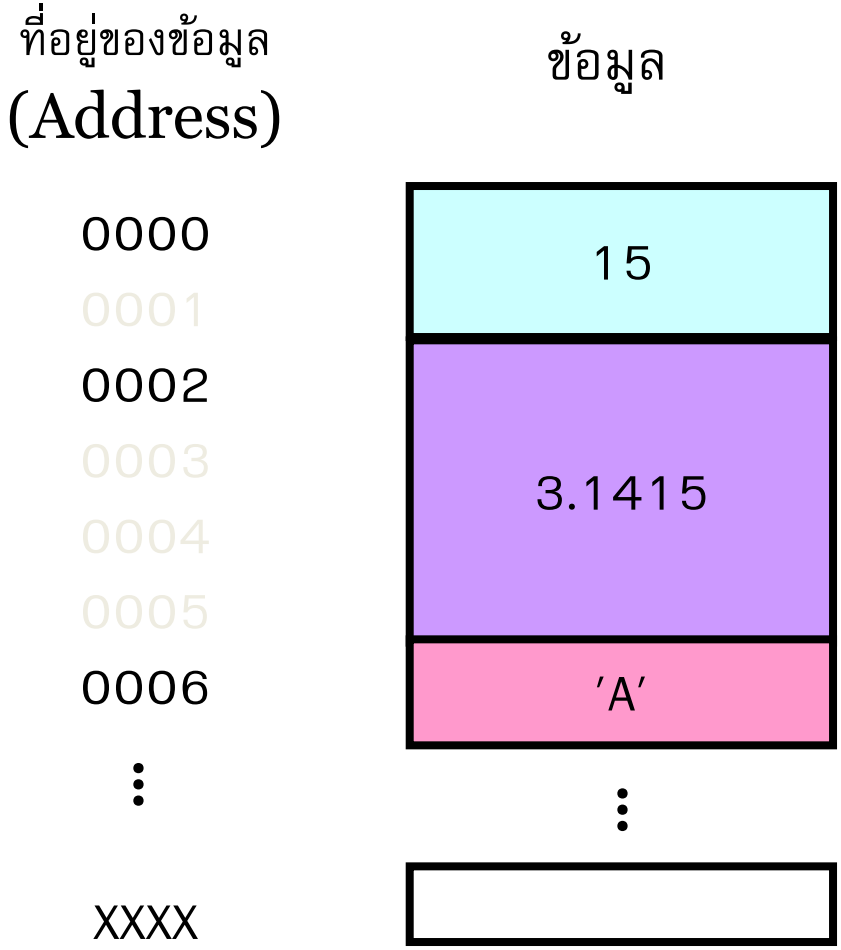
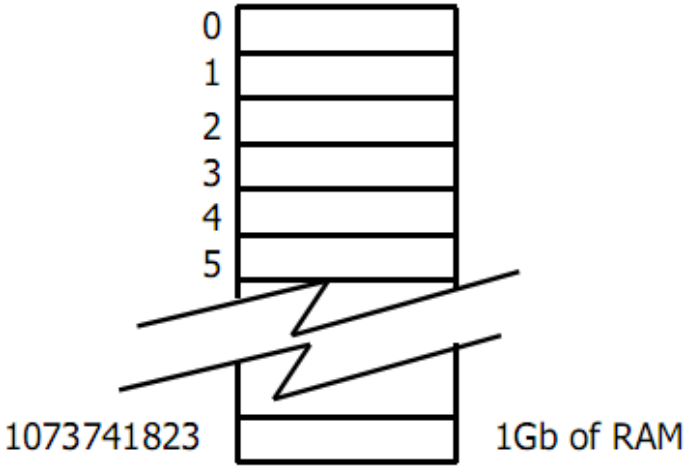
Physical Computing

RAM (หน่วยความจำ)

ประกอบด้วย ตำแหน่ง (Address) กับ ค่าข้อมูลที่เก็บ (Values)





1. บทนำ

ตัวอย่าง เมื่อประกาศคำสั่ง

```
int x=2;
```

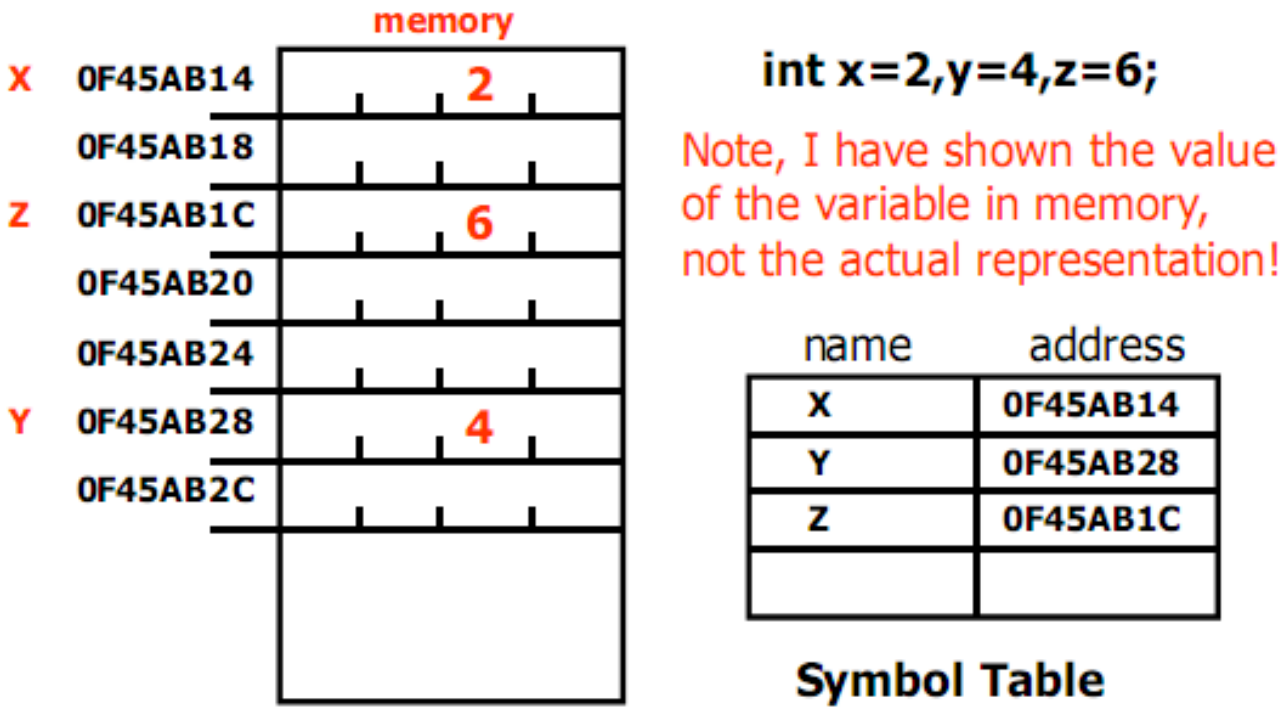
การทำงานที่เกิดขึ้น จะเกิดขึ้นดังนี้

- จองเนื้อที่ใน RAM เพื่อเก็บค่าตัวเลขจำนวนเต็ม
- นำชื่อตัวแปร **x** ไปเป็นชื่อ ตำแหน่ง address ของ RAM
- จัดเก็บค่าข้อมูล **2** ลงไปในตำแหน่งนั้น

x คือ Location name

2 คือ Values at Location

0F45AB14 คือ Address number (Location Number)



- จากรูป อธิบายได้ว่า

Computer เลือก Address ที่ 0F45AB14 ของหน่วยความจำ ในการเก็บข้อมูลเลข 2 ซึ่งบางครั้งอาจจะเป็น Address อื่นๆ ที่ว่างอยู่ก็ได้ ไม่จำเป็นต้องเป็นหมายเลข Address หมายเลข 0F45AB14

2. การแสดงตำแหน่ง address

เราสามารถแสดงตำแหน่ง address ที่เก็บข้อมูลได้ดังนี้

```
#include <stdio.h>
int main()
{
int a=3;
   printf("Address of a = %d \n", &a);
   printf("Values of a = %d \n",a);
}
```

```
Lecturer01@panwit ~
$ ./a
Address of a = 2280740
Values of a = 3
```

& มีความหมายว่า Address of

& อ่านว่า แอมเพอร์แซนด์ (ampersand) หรือเรียกโดยทั่วไปว่า **แอนด์** (and)

2. การแสดงตำแหน่ง address

Note: %d แสดงตำแหน่งเป็นเลขฐานสิบ

%p แสดงตำแหน่งเป็นเลขฐานสิบหก

```
#include <stdio.h>
int main()
{
   int a=3;
   printf("Address of a = %d \n", &a);
   printf("Address of a = %p \n", &a);

}
```

```
Lecturer01@panwit ~
$ ./a
Address of a = 2280740
Address of a = 0x22cd24
```

3. การแสดงค่าข้อมูล ที่ address นั้น ๆ

เราจะใช้เครื่องหมาย * หมายความว่า Values at Address

โดยที่ * จะวิ่งไปยัง address นั้นๆ เพื่อนำค่าออกมาแสดง ดังตัวอย่าง

```
#include <stdio.h>
int main()
{
   int a=3;
   printf("Address of a = %d \n", &a);
   printf("Values of a = %d \n", a);
   printf("Values of a = %d \n", *(&a) );
}
```

```
Lecturer01@panwit ~
$ ./a
Address of a = 2280740
Values of a = 3
Values of a = 3
```

* **อ่านว่า แอสเทอริส** (asterisk) หรือเรียกโดยทั่วไปว่า **ดอกจัน**

4. การประกาศตัวแปร pointer

- จากตัวอย่างที่ผ่านมา คำสั่ง **&a** ใช้แสดง address ของตัวแปร **a**

เราสามารถนำค่า address ดังกล่าวไปเก็บในตัวแปรได้เช่นกัน เช่น

```
b = &a
```

จะเห็นว่าตัวแปร **b** จะเก็บค่า Address ของตัวแปร **a**

แต่ในการเขียนโปรแกรม ต้องประกาศตัวแปร **b** ก่อน ถึงจะใช้งานได้

และเนื่องจาก **b** ใช้เก็บค่า address ดังนั้น เราสามารถประกาศตัวแปร **b** ได้เป็น

```
int  *b
```

เรียกตัวแปร **b** ว่าเป็นตัวแปรชนิด pointer

ตัวอย่างการใช้ตัวแปร pointer

```c
#include <stdio.h>
int main()
{
int a=3;
int *b;
b = &a;

printf("Address of a = %d \n", &a);
printf("Address of a = %d \n", b);
printf("Address of b = %d \n", &b);

printf("Values of a = %d \n", a);
printf("Values of a = %d \n", *(&a) );
printf("Values of a = %d \n", *b );

printf("Values of b = %d \n", b );

}
```

```
Lecturer01@panwit ~
$ ./a
Address of a = 2280740
Address of a = 2280740
Address of b = 2280736
Values of a = 3
Values of a = 3
Values of a = 3
Values of b = 2280740
```

5. รูปแบบการประกาศตัวแปร pointer

```
type   *pointer_name
```

type คือ ชนิดของตัวแปร pointer โดยพิจารณาว่าจะใช้เก็บ address ของตัวแปรชนิดใด

***** คือ เครื่องหมายแสดงว่าเป็นตัวแปรประเภท pointer

pointer_name คือ ชื่อของตัวแปรประเภท pointer

ตัวอย่างการประกาศตัวชี้

```
int         num;
int         *pnum;
float       salary = 1200.50;
float       *psalary ;
```

ตัวอย่าง

```c
#include<stdio.h>
int main()
{
  char          letter = 'D';
  int           num    = 19;
  float point   = 26.09;
  char          *pt_letter;
  int           *pt_num;
  float         *pt_point;
  pt_letter = &letter;
  pt_num    = &num;
  pt_point  = &point;

  printf("Address of letter =%p \n",pt_letter);
  printf("Address of num    =%p \n",pt_num);
  printf("Address of point  =%p \n",pt_point);
  return 0;
}
```

```
Lecturer01@panwit ~
$ ./a
Address of letter = 0x22cd27
Address of num    = 0x22cd20
Address of point  = 0x22cd1c
```

6. pointer ของ pointer

จากที่ผ่านมา pointer ก็คือตัวแปรที่ใช้ในการเก็บ address ของตัวแปรอื่นๆ ดังนั้น pointer ก็สามารถเก็บ address ของ pointer ได้เช่นกัน เรียกอีกอย่างว่า nested pointer

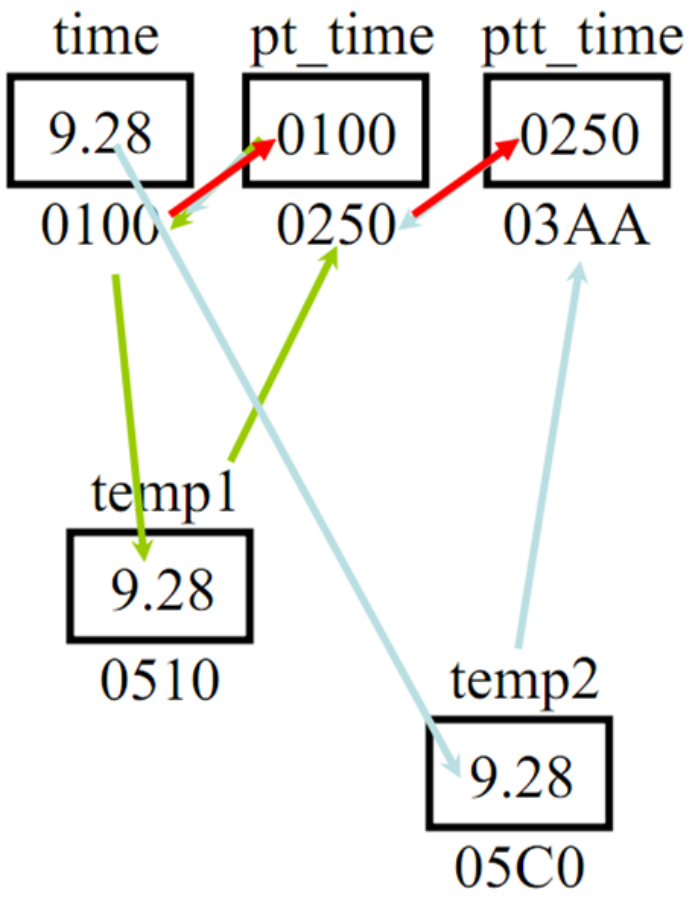
ตัวอย่าง

```c
#include <stdio.h>
int main()
{
    int a=3 , *b , **c;
    b = &a;
    c = &b;
    printf("Address of a = %d \n", &a);
    printf("Address of a = %d \n", b);
    printf("Address of a = %d \n", *c);
    printf("Address of b = %d \n", &b);
    printf("Address of b = %d \n", c);
    printf("Address of c = %d \n", &c);
    printf("Values of a = %d \n", a);
    printf("Values of a = %d \n", *(&a) );
    printf("Values of a = %d \n", *b );
    printf("Values of a = %d \n", **c );
    printf("Values of b = %d \n", b );
    printf("Values of c = %d \n", c );
}
```

```
Lecturer01@panwit ~
$ ./a
Address of a = 2280740
Address of a = 2280740
Address of a = 2280740
Address of b = 2280736
Address of b = 2280736
Address of c = 2280732
Values of a = 3
Values of a = 3
Values of a = 3
Values of a = 3
Values of b = 2280740
Values of c = 2280736
```

ตัวอย่าง Nest Pointer

```
float time = 9.28;
float *pt_time;
float **ptt_time;
pt_time = &time;
ptt_time = &pt_time;
float temp1;
temp1 = *pt_time;
float temp2;
temp2 = **ptt_time;
```



Work to your Brain

```c
#include <stdio.h>
int main(void)
{
	int a;
	int *p;
	int **q;
	a = 14;
	p = &a;
	q = &p;
	printf("%d\n", a);
	printf("%d\n", p);
	printf("%d\n", q);
	printf("%d\n", &a);
	printf("%d\n", &p);
	printf("%d\n", &q);
	printf("%d\n", *p);
	printf("%d\n", *q);
	printf("%d\n", **q);
	return 0;
}
```

กำหนดให้

	Address	value
a		14
p		2280740
q		2280736

เมื่อ run program จะได้ผลลัพธ์อย่างไร

.....

.....

.....

.....

.....

.....

.....

.....

.....

Work to your Brain

```
int x;
int *px;
int **py;

px = &x;
py = &px;
*px = 5;
printf(“%d\n",x);
printf("%d\n",*px);
printf("%p\n",px);
printf("%p\n",&x);
printf("%d\n",*(&x));
printf("%p\n",&px);
printf("%p\n",*py);
```

กำหนดให้

	Address	value
x		5
px		0xf7f8a4
py		0xf7f89c

เมื่อ run program จะได้ผลลัพธ์อย่างไร

.....

.....

.....

.....

.....

.....

.....

7. การกระทำทางคณิตศาสตร์ ของ Pointer

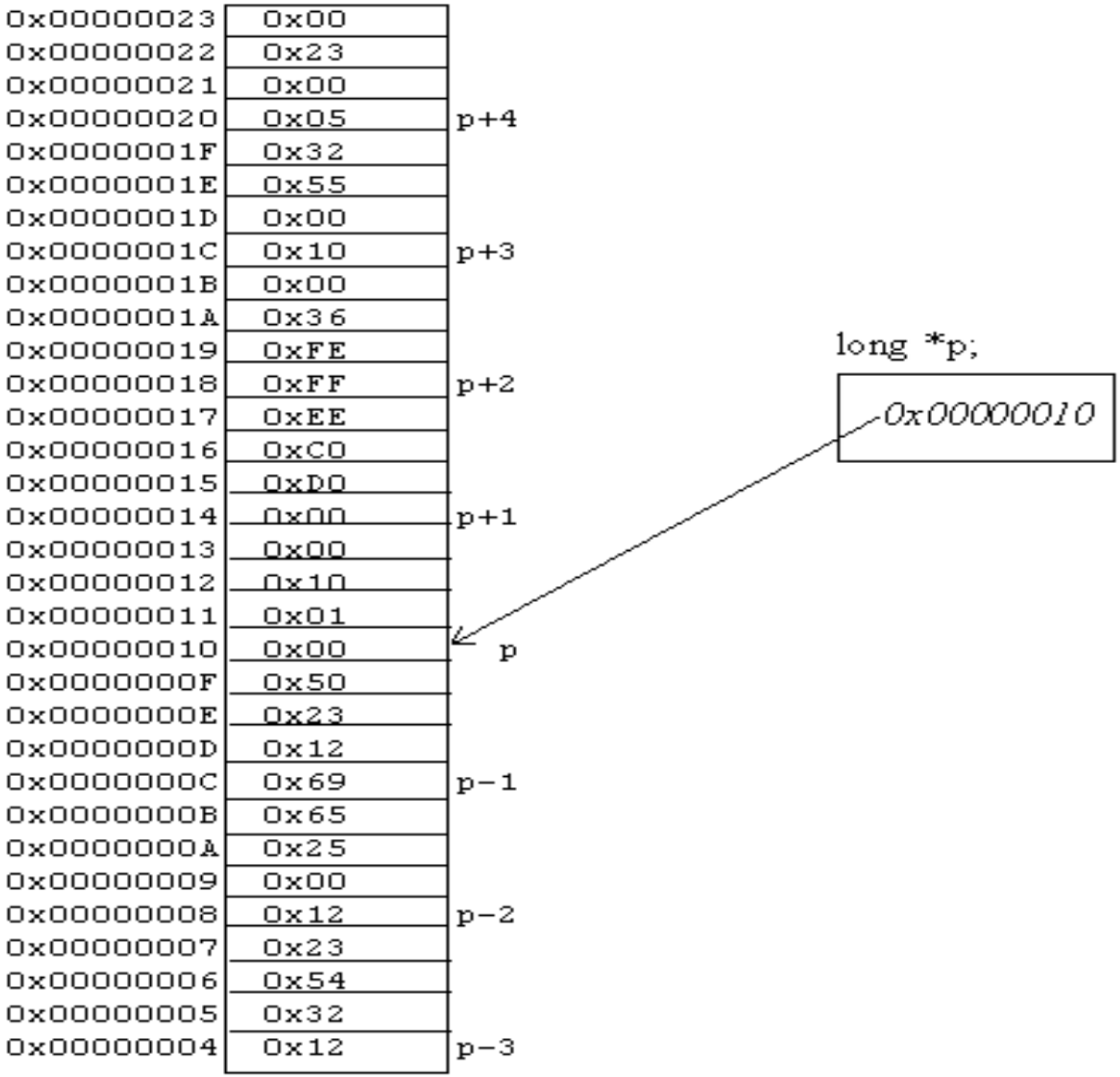
- เครื่องหมายทางคณิตศาสตร์ที่ใช้กับ pointer ได้ จำกัดอยู่เพียง +,-,++,-- โดยการ + คือการเพิ่มค่า address ให้สูงขึ้น และ การ – คือ การลดค่า address ให้ต่ำลง

```c
#include <stdio.h>
int main()
{
   int a=3;
   int *b;
   b = &a;
   printf("Address of a = %d \n", b);
   b=b+1;
   printf("Address of a = %d \n", b);
   b=b+2;
   printf("Address of a = %d \n", b);
}
```

```
Lecturer01@panwit ~
$ ./a
Address of a = 2280740
Address of a = 2280744
Address of a = 2280752
```

- sizeof(int) = 4
- sizeof(long) = 4
- sizeof(long int) = 4
- sizeof(long long int) = 8
- sizeof(char) = 1
- sizeof(float) = 4
- sizeof(double) = 8
- sizeof(long double) = 12

สังเกตถึงการบวก
หรือลบค่าแอดเดรส
ด้วยค่าจำนวนเต็ม
ทำให้เกิดการเลื่อน
ไปของค่าแอดเดรส
ผลลัพธ์ ขึ้นอยู่กับ
ชนิดของตัวชี้



8. ความสัมพันธ์ของ Pointer และ Array

เมื่อใช้ Array และมีการประกาศ Pointer ของ Array
Pointer จะชี้ไปที่ Address index แรก ของ Array

ในการพิมพ์ค่าของตัวแปร Array แต่ละตัว ทำได้โดย เพิ่มค่า Address ในตัวแปร Pointer หรือเลื่อน Pointer ไปยัง Address ที่สูงขึ้น ซึ่งอยู่ติดกัน

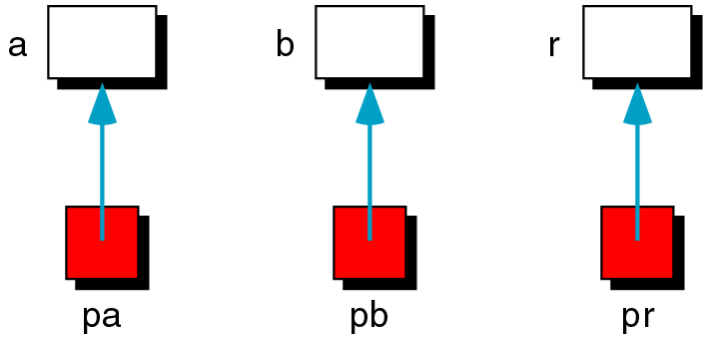
8. ความสัมพันธ์ของ Pointer และ Array

```c
#include <stdio.h>
int main()
{
   int number[] = {100,200,300};
   int *pt;
   pt = number;
   printf("number[0] = %d \n", *pt);

   printf("number[1] = %d \n", *(pt+1));

   printf("number[2] = %d \n", *(pt+2));
}
```

```
/tmp/5caoHaNKOW.o
number[0] = 100
number[1] = 200
number[2] = 300
```

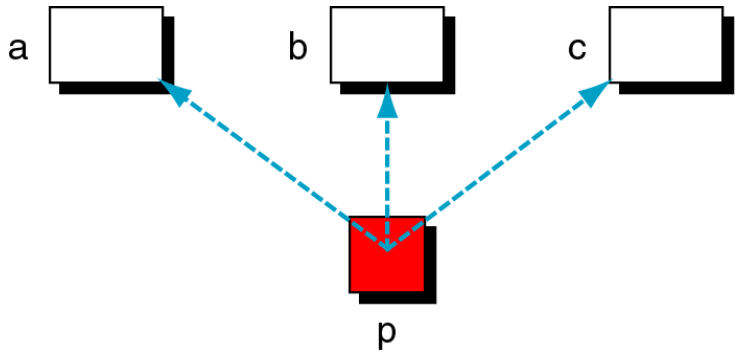
Fun with a pointer : Add two numbers using pointers

```c
#include <stdio.h>
int main()
{
    int a, b, r;
    int *pa = &a;
    int *pb = &b;
    int *pr = &r;

    printf("Enter the first number: ");
    scanf("%d", pa);

    printf("Enter the second number: ");
    scanf("%d", pb);

    *pr = *pa + *pb;

    printf("\n%d + %d is %d", *pa, *pb, *pr);
    return 0;
}
```



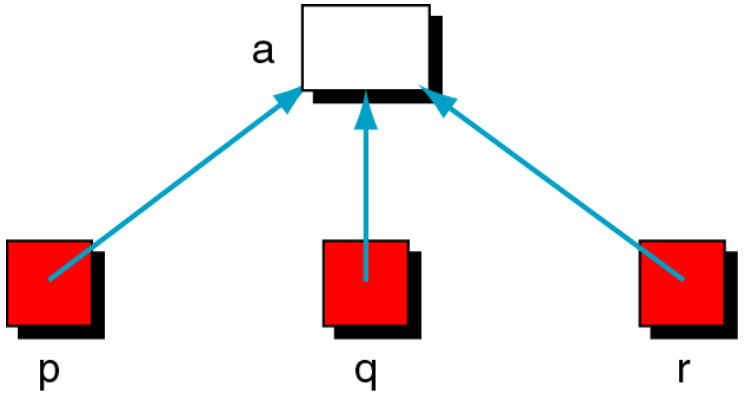
Fun with a pointer :
Using one pointer for many variables

```
#include <stdio.h>
void main (void)
{
        int   a, b, c;
        int *p;

        printf("Enter three numbers and key return: ");
        scanf("%d %d %d", &a, &b, &c);
        p = &a;
        printf("\n %3d ", *p);
        p = &b;
        printf("\n %3d ", *p);
        p = &c;
        printf("\n %3d ", *p);
}
```



Fun with a pointer : Using a variable with many pointers

```
#include <stdio.h>
void main (void)
{
        int   a;
        int *p = &a;
        int *q = &a;
        int *r = &a;

        printf("Enter a number: ");
        scanf("%d", &a);
        printf("\n %d ", *p);
        printf("\n %d ", *q);
        printf("\n %d ", *r);
}
```



Function: pass by value

```
void myFunc(int arg);

int main(void)
{
  int x = 5;
  myFunc(x);
  printf("%d\n", x);
}

void myFunc(int arg)
{
  arg = 4;
}
```

```
/* Output */
5
```

Function: pass by reference

```
void myFunc(int *arg);

int main(void)
{
  int x = 5;
  myFunc(&x);
  printf("%d\n", x);
}

void myFunc(int *arg)
{
  *arg = 4;
}
```

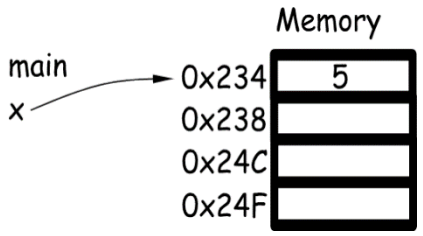
```
/* Output */
4
```

Function: pass by value

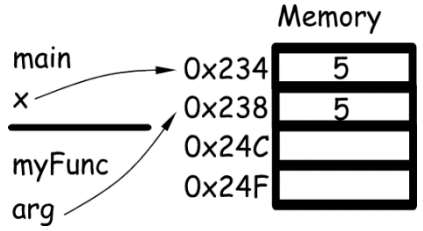
```
void myFunc(int arg);

int main(void)
{
  int x = 5;
  myFunc(x);
  printf("%d\n", x);
}

void myFunc(int arg)
{
  arg = 4;
}
```

```
/* Output */
5
```

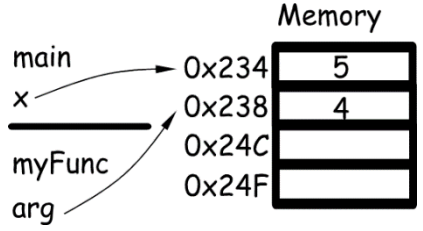
int x = 5



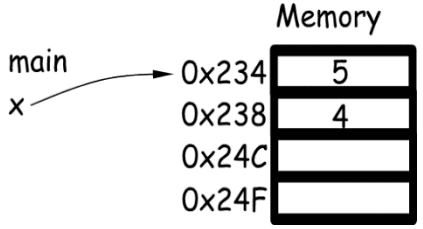
myFunc(x)



arg = 4



printf("%d\n",x)

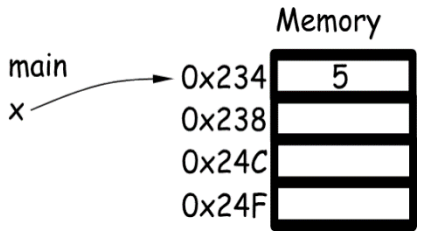


Function: pass by reference

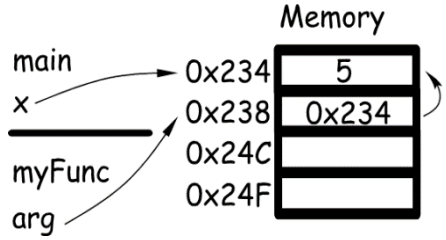
```
void myFunc(int *arg);

int main(void)
{
  int x = 5;
  myFunc(&x);
  printf("%d\n", x);
}

void myFunc(int *arg)
{
  *arg = 4;
}
```

```
/* Output */
4
```

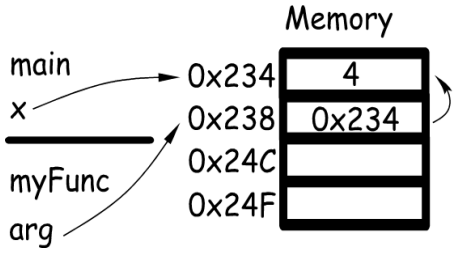
int x = 5



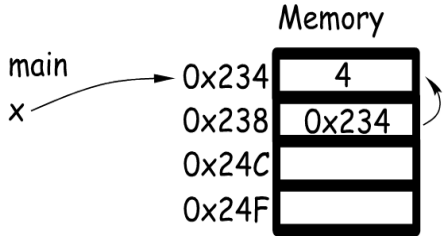
myFunc(&x)



***arg = 4**



printf("%d\n",x)



Work to your Brain

```c
#include <stdio.h>
void swap1(int a, int b)
{
    int temp ;
    temp =a;
    a = b;
    b = temp;
}

void swap2(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
int main()
{
    int x = 10;
    int y = 20;
    printf("Start: x = %d, y = %d\n", x, y);
    swap1(x, y);
    printf("swap1: x = %d, y = %d\n", x, y);
    swap2(&x, &y);
    printf("swap2: x = %d, y = %d\n", x, y);
    return 0;
}
```

เมื่อ run program

จะได้ผลลัพธ์อย่างไร

จงอธิบายการทำงาน ว่าเหตุใด

ค่าที่ได้จาก swap1

และ swap2 ถึงมีค่าไม่เท่ากัน

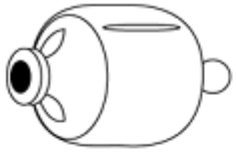
แบบฝึกหัด

1. เขียนโปรแกรมเพื่อรับข้อมูลตัวเลขทศนิยมสามจำนวน แล้วส่งเลขทศนิยมทั้งสามนี้ไปยังฟังก์ชั่นย่อย
โดยให้ฟังก์ชั่นย่อย ทำการสลับตำแหน่งของตัวเลขทั้งสาม
โดยให้ตัวแรกไปอยู่ที่ตำแหน่งที่สอง ตัวเลขค่าที่สองย้ายไปอยู่ตำแหน่งที่สาม ตัวเลขค่าที่สามย้ายไปอยู่ตำแหน่งแรก
เช่น **input 5.12 9.26 6.99 output 6.99 5.12 9.26**

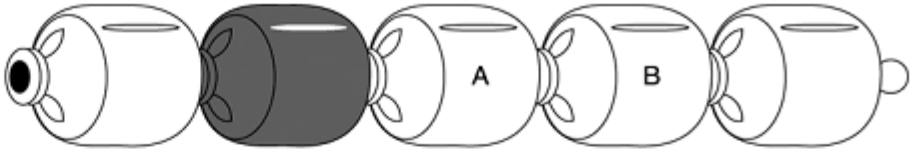
2. เขียนโปรแกรมรับข้อความ แล้วตรวจสอบว่าเป็นข้อความสมมาตร (palindrome) หรือไม่ โดยใช้ pointer
เช่น **MADAM , HANNAH , Never odd or even**

3. เขียนโปรแกรมรับค่า แล้วแสดงผลแบบย้อนกลับ
เช่น **input : I love programming**
output: gnimmargorp evol I

ตัวอย่างการประยุกต์ : Link list



Pop bead

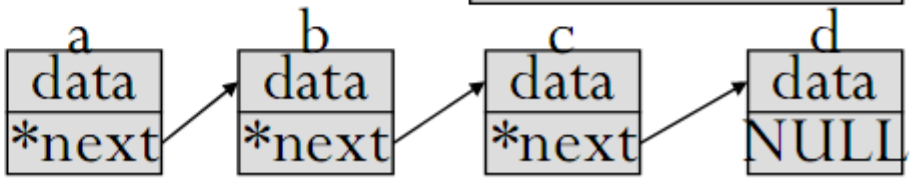


Chain of pop beads

- A linked list is a sequence of nodes in which each node but the last contains the address of the next node.
- An alternative to arrays.
- Common abstraction to use in programming.

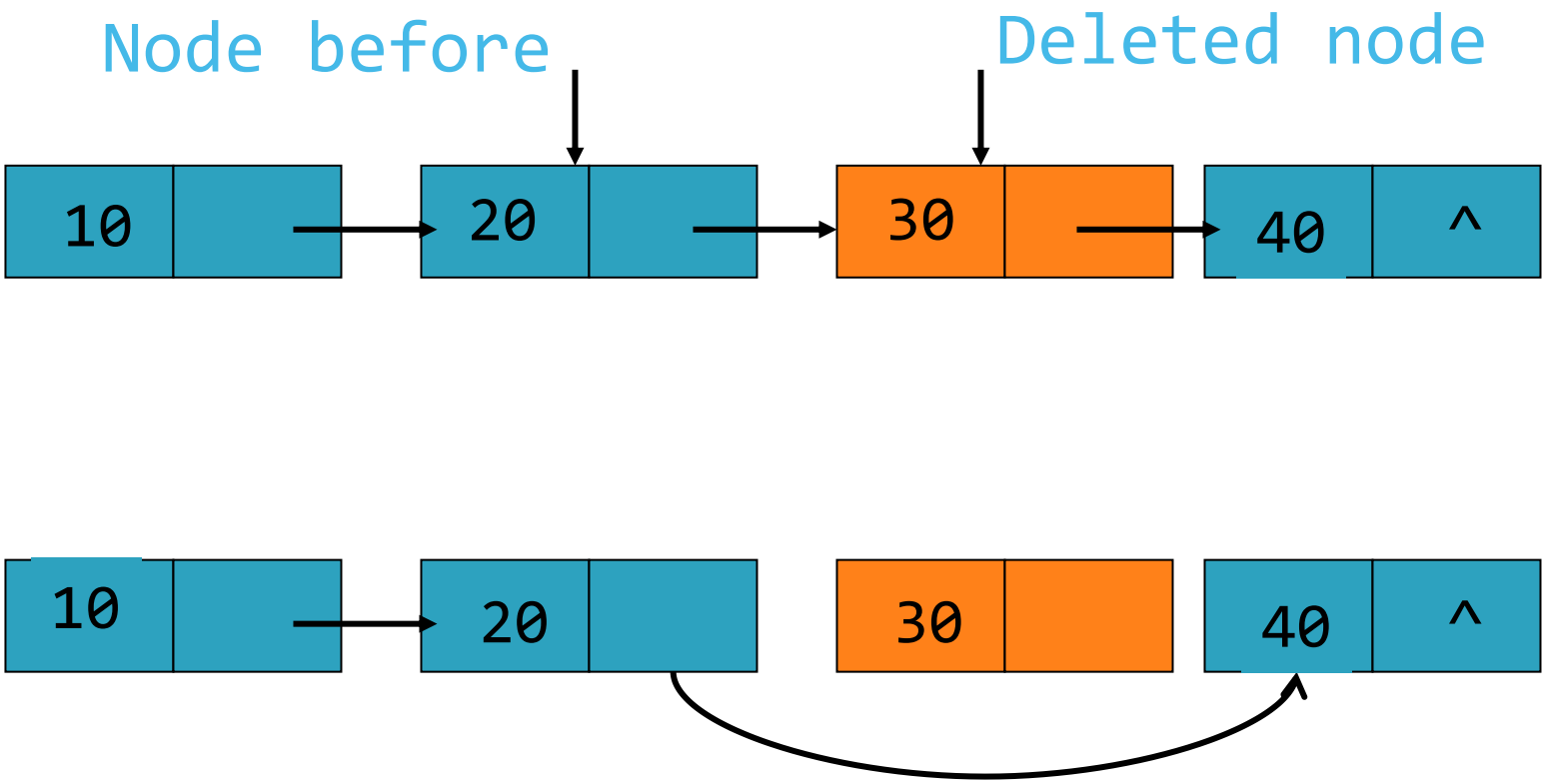
```
a.next = &b;
b.next = &c;
a.next -> next -> next  = &d;
/* an alternative c.next = &d; */
d.Next = NULL;
```

```
struct list {
   int data;
   struct list *next;
} a, b, c, d;
```



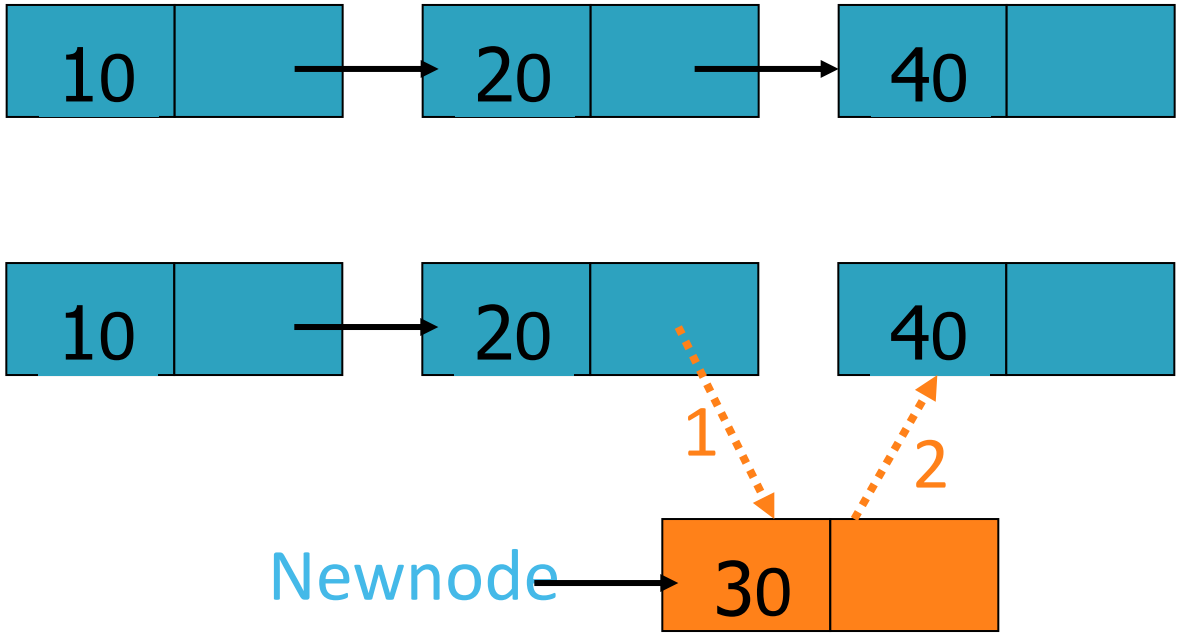
2. การลบ Node ใน Link list

- การลบ Node ทำได้โดยการเปลี่ยน Pointer ของ Node ก่อนที่จะลบไปชี้ Node ของตัวที่ถัดจาก Node ที่ได้ลบไป



3. การเพิ่ม Node เข้าไปใน Link list

- เป็นการแทรก New Node เข้าไปใน Link List
- ก่อนจะแทรกจะต้องทราบตำแหน่งที่จะแทรก
- เปลี่ยน Pointer ของ Node ก่อนหน้า ไปยัง Address ของ New Node
- และให้ Pointer ของ New Node ชี้ไปยัง Address ของ Node ตัวถัดไป




ANY
QUESTIONS?