344-111

CONCEPT OF POINTERS

Outline

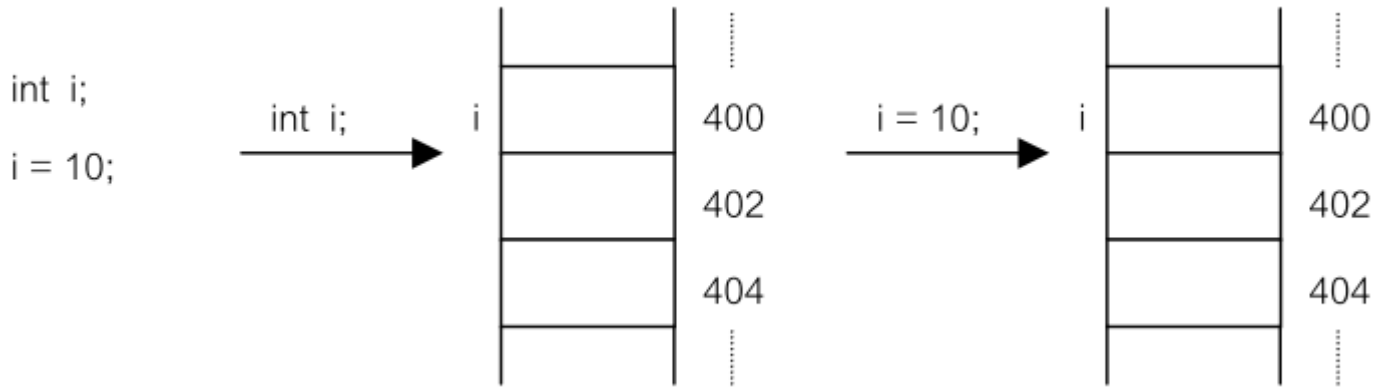
- What are pointer?
- Pointers and address
- Pointer declarations
- How to use pointers
- Pointer Arithmetic
- Using pointer with array
- Pointer to Array
- Array of Pointer

What are Pointers?

- Pointers เป็นพื้นฐานการเขียนโปรแกรมในภาษา C หรือ C++
- Pointers มีความเกี่ยวข้องกับ Address ของหน่วยความจำ
- Pointers เป็นวิธีหนึ่งในการอ้างถึงตัวแปรอื่น โดยใช้ address ของตัวแปรนั้นๆ แทนที่จะเก็บค่าทั้งหมดไว้
- มีการใช้งานที่อ้างอิงถึง Address ของหน่วยความจำ ในเรื่องของ Array
  - การเรียกชื่อตัวแปร array เป็นการแทนด้วย Address ของหน่วยความจำ

Idea of Pointers

- โดยทั่วไป ตัวแปรที่ประกาศขึ้นมาจะมีการจองพื้นที่ในหน่วยความจำไว้เพื่อเก็บ specific value
  - เช่น การประกาศตัวแปรชื่อ i เป็นตัวแปรประเภท int
  - สมมุติให้ชนิดข้อมูล Integer มีขนาด 1 ไบต์
  - การแทนข้อมูลในหน่วยความจำของตัวแปรเป็นดังนี้



Address in C

- เมื่อเราสั่งให้ทำงาน int i=7 จะทำให้ complier ไปจองพื้นที่ใน Memory ที่มีขนาดเท่ากับประเภทของตัวแปรนั้น (ขนาดของตัวแปรแต่ละประเภทในภาษาซีขึ้นอยู่กับบริษัทผู้ผลิตคอมไพเลอร์)
- เช่น จอง memory address ที่ 400 สำหรับเก็บค่า 7

```c
#include<stdio.h>

void main()
{
    int var = 5;
    printf("Value of the variable var is: %d\n", var);
    printf("Memory address of the variable var is: %x\n", &var);
}
```

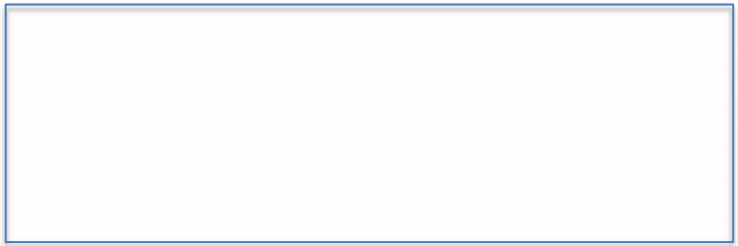
```
Value of the variable var is: 5
Memory address of the variable var is: 28ff1c
```

Address

- การอ้างถึง address ของตัวแปรในภาษาซี ใช้เครื่องหมาย & (ampersand) นำหน้าตัวแปร

```
#include <stdio.h>

void main()
{
        int x;

        printf("%d %d\n", &x, x);

        x=7;

        printf("%d %d\n", &x, x);

}
```

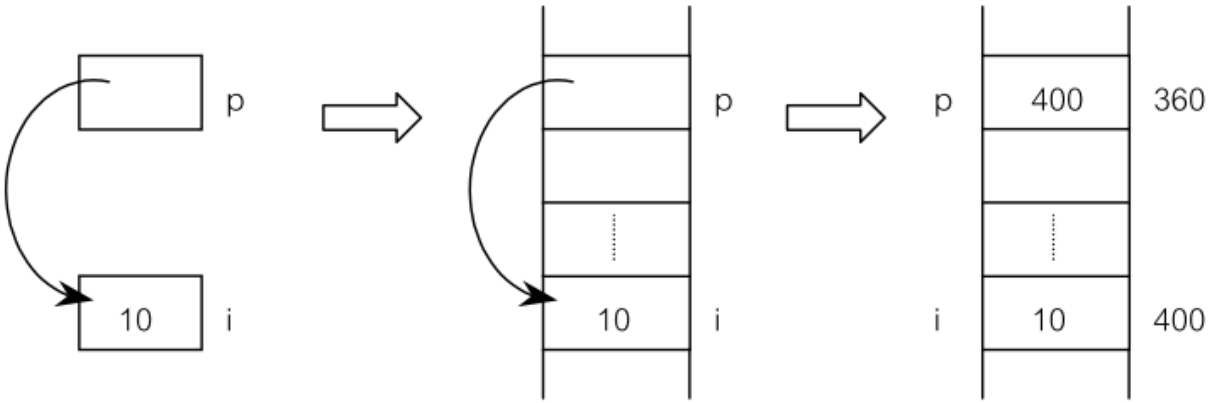
ผลลัพธ์ทางจอภาพ



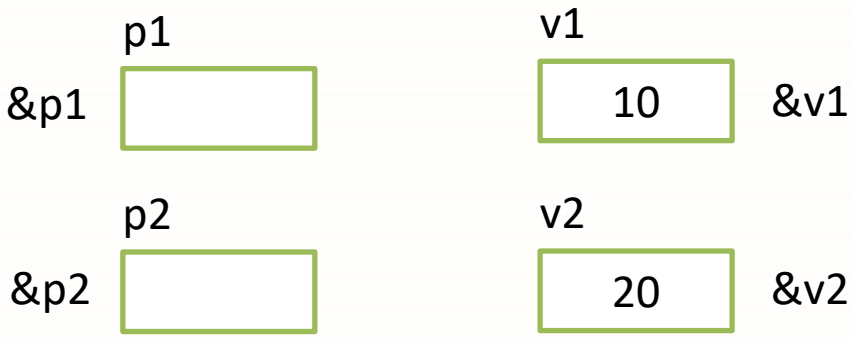
สรุปได้ว่า

Idea of Pointers

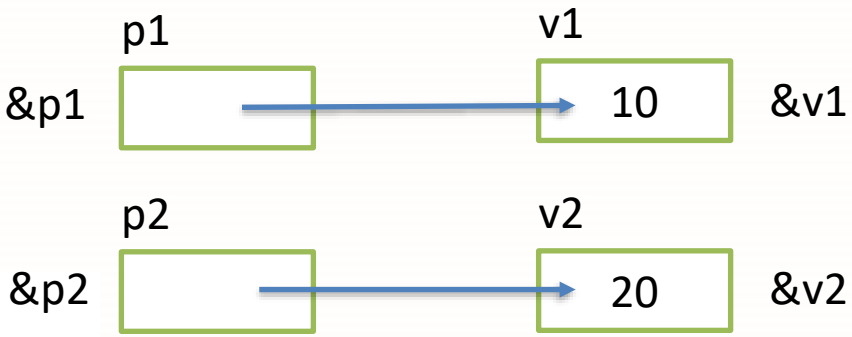
- ตัวแปรพอยน์เตอร์เป็นตัวแปรที่ใช้เก็บค่าแอดเดรสของตัวแปรอื่นๆ ตัวอย่างเช่น
  - i เป็นตัวแปรประเภท int
  - ตัวแปร p เป็นตัวแปรประเภทพอยน์เตอร์ที่เก็บค่าแอดเดรสของตัวแปร i (หรือ p ชี้ไปที่ตัวแปร i)
  - สามารถจำลองการแทนข้อมูลในหน่วยความจำได้ดังนี้



Example

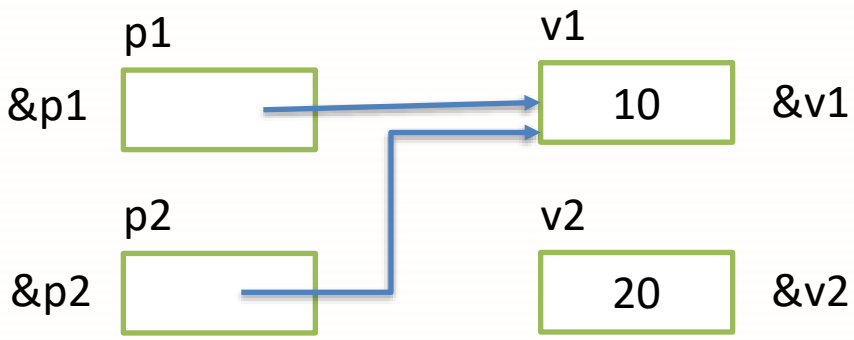


p1 = &v1
p2 = &v2



Example

p2 = p1



ถ้ามีการเปลี่ยนค่า **v1** เป็น 30 มีกี่ตัวแปรที่เปลี่ยนค่าด้วย **??**

Pointer declarations

- ตัวแปรทั่วไปประกาศขึ้นเพื่อ....
- ตัวแปรพอยน์เตอร์ประกาศเพื่อ...

- รูปแบบการประกาศ

```
type_var        *pt_name
```

```
type_var*       pt_name
```

type_var คือ ประเภทข้อมูลตัวแปร เช่น int, float, char

* คือ เป็นเครื่องหมายเพื่อกำหนดให้ตัวแปรที่ประกาศเป็น pointer

pt_name คือ ชื่อตัวแปร

Pointer declarations

- ตัวอย่างการประกาศตัวแปรพอยน์เตอร์

- int *p1;
  - สำหรับเก็บตำแหน่งที่อยู่ของตัวแปรชนิด int เท่านั้น
- char *p2;
  - สำหรับเก็บตำแหน่งที่อยู่ของตัวแปรชนิด char เท่านั้น
- float *p3;
  - สำหรับเก็บตำแหน่งที่อยู่ของตัวแปรชนิด float เท่านั่น

Pointer declarations

- Pointers ควรมีการกำหนดค่าเริ่มต้น โดยสามารถกำหนดเป็น 0, null หรือ address
  - Pointer มีค่าเป็น null หมายถึงว่า pointer ไม่ได้ชี้ไปที่ใด
  - Pointer มีค่าเป็น 0 หมายถึงว่า pointer ไม่ได้ชี้ไปที่ใด
  - แต่ควรใช้ null เพราะว่าเมื่อมีการกำหนดด้วย 0 และเมื่อมีการกำหนดค่าครั้งแรก ต้องมีการแปลงให้เป็นชนิดข้อมูลที่เหมาะสม

- การสร้างตัวแปรมีสองแบบ
  - Static allocated memory
  - Dynamic allocated memory จะกล่าวในภายหลัง

How to Use Pointers?

- การกำหนดค่าให้กับ pointer
  - การกำหนดค่าให้กับตัวแปร pointer เป็นการกำหนด address ของตัวแปรที่มีชนิดข้อมูลสอดคล้องกับตัวแปรพอยน์เตอร์นั้น
  - ใช้เครื่องหมาย & สำหรับการอ้างถึง address

```
pt_name = &var;
```

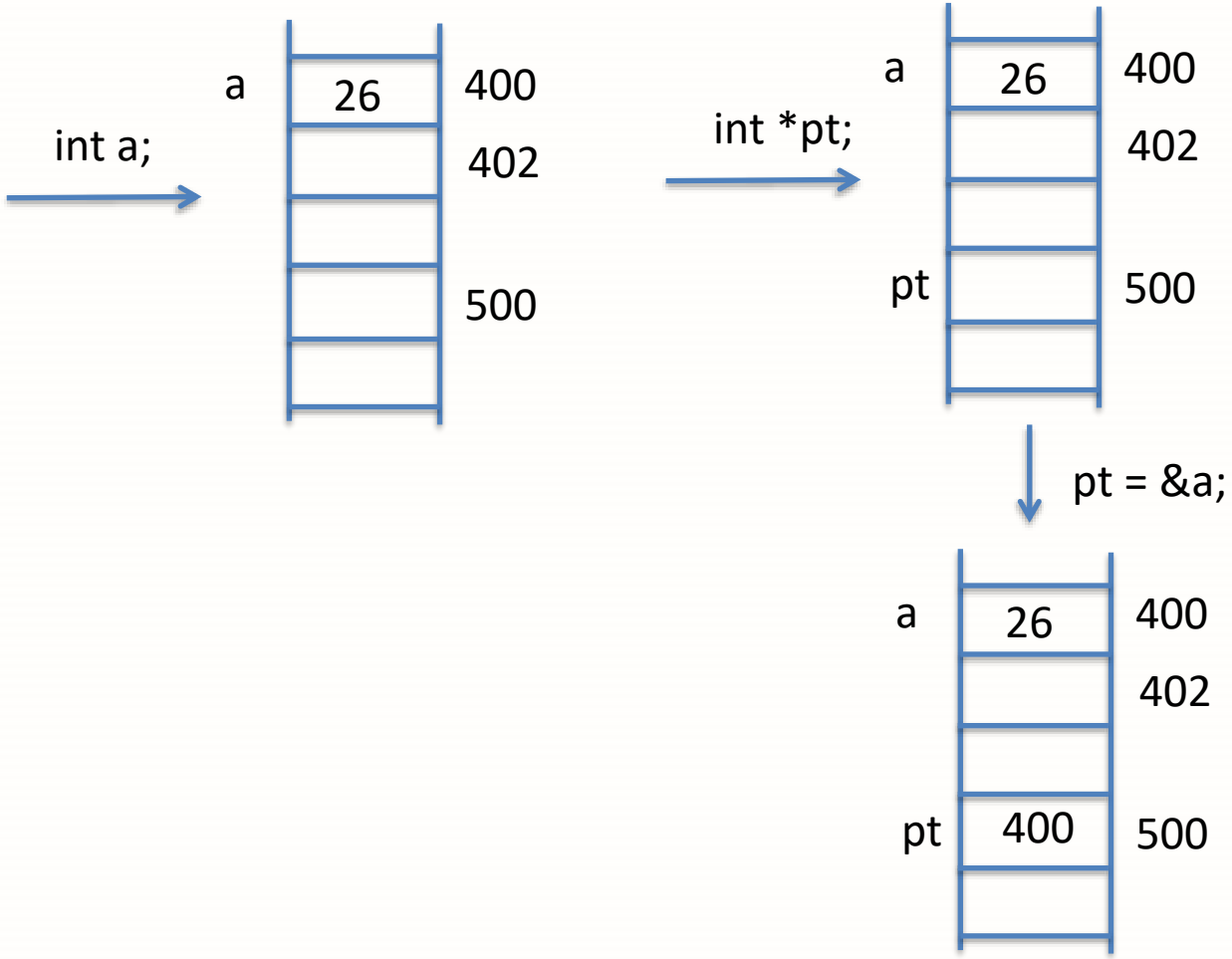
pt_name เป็นชื่อตัวแปรพอยน์เตอร์

var ชื่อตัวแปรที่จะส่ง address ไปเก็บไว้ที่ตัวแปร pt_name

How to Use Pointers?

- ตัวอย่าง

```
int  a = 26;
int  *pt;
pt  =  &a;
```



How to Use Pointers?

Try to run and compile the following program

```
int   a = 26;
int  *pt;
pt  =  &a;

printf("\na=%d \t&a=%d\n",a, &a);

printf("\npt=%d \t&pt=%d\n",pt, &pt);
```

How to Use Pointers?

- สามารถอธิบายคำสั่งแต่ละบรรทัดได้ดังนี้

int a = 26;

เป็นการประกาศตัวแปร a โดยกำหนดให้มีชนิดเป็น int และกำหนดค่าเริ่มต้นให้ตัวแปร a เป็น 26

int *pt;

ประกาศตัวแปรชื่อว่า pt เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท int

pt = &a; (อ่านว่า pt ชี้ไปยัง a)

กำหนดค่า address ของตัวแปร a ให้กับตัวแปรพอยน์เตอร์ pt

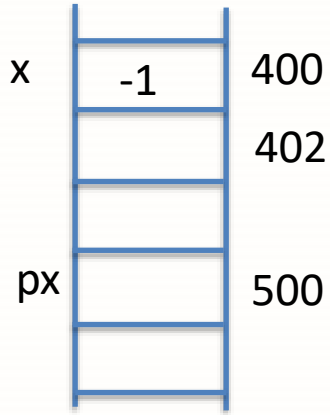
How to Use Pointers?

- การอ้างอิงค่าข้อมูลบนตำแหน่งที่อยู่ (Indirect operator: *)
  - (star) ใช้นำหน้าตัวแปร เมื่อต้องการอ้างอิงค่าที่เก็บอยู่ใน address ที่ตัวแปรนั้นเก็บค่าอยู่
  - ตัวอย่าง

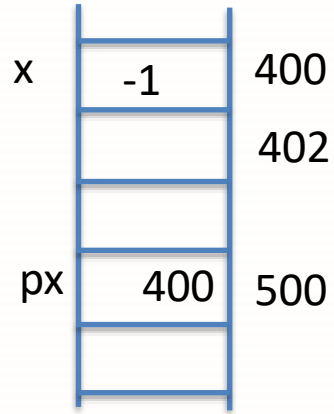
Example

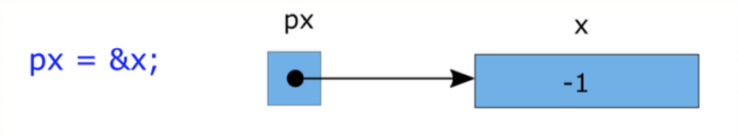
```
float x  = -1;
float *px;
px  = &x;
*px  =  3.14;
x   =  2018
```

```
float x  = -1;
float *px;
```




```
px  = &x;
```

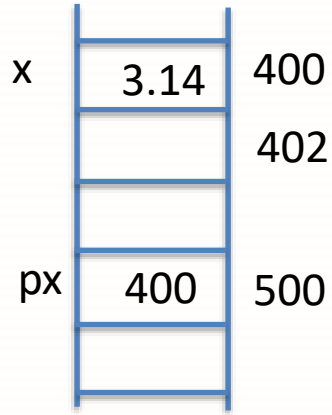


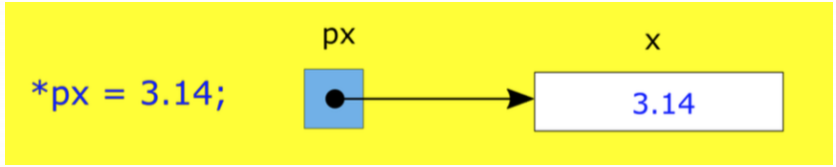


Example

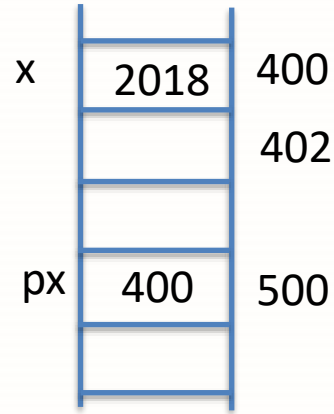
```
float x  = -1;
float *px;
px  = &x;
*px  =  3.14;
x   =  2018
```

*px = 3.14;





x = 2018;



Pointer Arithmetic

- Several arithmetic operations are performed on pointers
  - Adding and subtracting an integer to pointer
  - Adding and subtracting an integer to pointer
  - Comparing pointers

```
1  #include<stdio.h>
2  void main()
3  {
4       int x;
5       int y =1;
6       int *pt1,*pt2;
7
8       pt1 = &x;
9       pt2 = &y;
10
11      printf("pt1 = %d\n",pt1);
12      printf("pt2 = %d",pt2);
13
14      printf("\n-----------------\n");
15
16      pt1++;
17      pt2--;
18
19      printf("pt1 = %d\n",pt1);
20      printf("pt2 = %d",pt2);
21 }
```

```
pt1 = 2686740
pt2 = 2686736
```

```
pt1 = 2686744
pt2 = 2686732
```

Using Pointers

- Create **dynamic data structures**
- Handle **variable parameters** passed to functions
- **Access information stored in arrays**

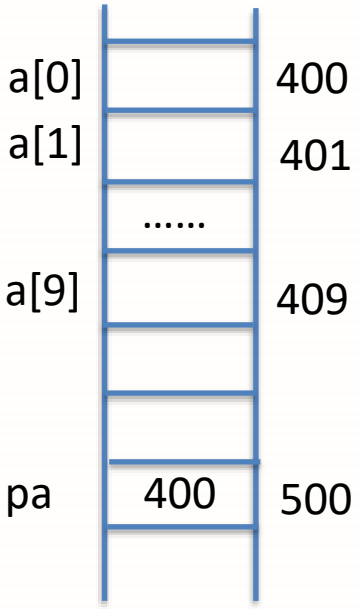
Pointers and Array

- Closely related in each other and interchangeably in the right context
- Useful when pointers work with array

- When array name is used by itself, the array's address is returned
- We can assigned this address to pointer

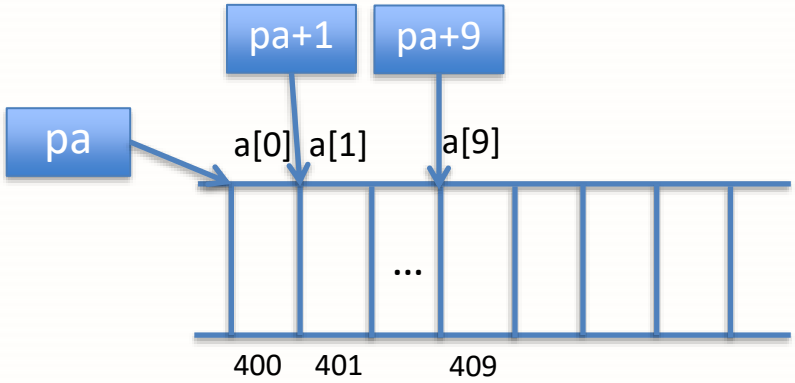
```
int a[10];
int  *pa = a;
```

pa is a pointer to the first element of array



Array and Pointers

- Consider the statements below, what are the results ?
  - printf("%d", a);
  - printf("%d", &a[0]);
- Question: pa = a and pa = &a[0] , are these two statements equivalent? What does it mean?
- using array subscripts with pointers
  - the notation a[i] = *(pa+i)



Array and Pointers

- Example

```c
#include<stdio.h>
int main()
{
    int a[10] = {2,3,4,5,6,7,8,9,0,1};
    int *pt;

    pt = &a;

    printf("pt,\t addres:%d \t value: %d\n",pt,*pt);
    printf("pt+1,\t addres:%d \t value: %d\n",pt+1,*(pt+1));
    printf("pt+3,\t addres:%d \t value: %d\n",pt+3,*(pt+3));

    return 0;
}
```

Output

```
pt,     addres:2686708          value: 2
pt+1,   addres:2686712          value: 3
pt+3,   addres:2686720          value: 5
```

Assignment 1

- Suppose that record is an array as the following

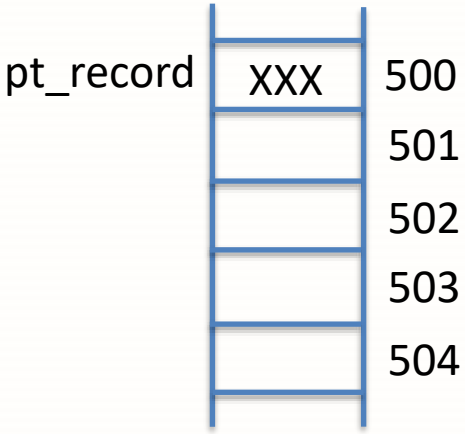
float record[5] = {32.46, 12.67, 43.908, 76.09, 12.401};

record[0]	32.4	0300
record[1]	12.67	0304
record[2]	43.908	0308
record[3]	76.09	030C
record[4]	12.401	0310

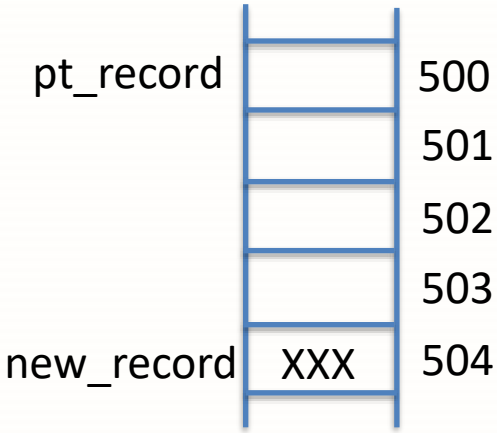
Assignment 1

What is XXX in each picture? Write your answer in LMS2@PSU

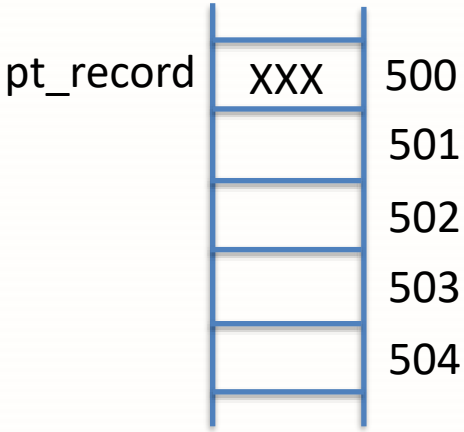
```
float  *pt_record;
pt_record = &record;
```




```
float  new_record;
new_record = *(pt_record+4);
```




```
pt_record = &record[2];
```



Assignment 2

- What is an output of the following program? Write your answer in LMS2@PSU

```c
#include<stdio.h>
int main ()
{
/* an array with 5 elements */
    double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
    double *p;
    int i;
    p = balance;
/* output each array element's value */
    printf( "Array values using pointer\n");
    for ( i = 0; i < 5; i++ )
    {
        printf("*(p + %d) : %f\n", i, *(p + i) );
    }

    printf( "Array values using balance as address\n");
    for ( i = 0; i < 5; i++ )
    {
        printf("*(balance + %d) : %f\n", i, *(balance + i) );
    }

    return 0;
}
```

Summary: Pointer and Array

- In the following program, a pointer *pt* points to the first element of array

```
1   #include<stdio.h>
2   int main()
3   {
4       int a[10] = {2,3,4,5,6,7,8,9,0,1};
5       int *pt;
6
7       pt = &a;
8
9       printf("pt,\t addres:%d \t value: %d\n",pt,*pt);
10      printf("pt+1,\t addres:%d \t value: %d\n",pt+1,*(pt+1));
11      printf("pt+3,\t addres:%d \t value: %d\n",pt+3,*(pt+3));
12
13      return 0;
14  }
```

Pointer and string

```c
#include<stdio.h>
int main()
{
  char *ptr = "Pointer-to-String", i;
  printf("%s", ++ptr);
  return 0;
}
```

Example

- เขียนโปรแกรมเพื่อคนหาจำนวนที่น้อยที่สุดในตัวแปรอาร์เรย์โดยใช้พอยเตอร์

```
#include<stdio.h>
main()
{
   int number[10] = {3, 1, 10, 20, 4, 30, 8, 12, 2, 0};
   int *p;
   int i,min;
   p = number;
   min = *p;

   for(i =0;i<10;i++)
   {
      if(*(p+i) <= min)
         min = *(p+i);
   }

   printf("%d",min);
}
```

Dynamic Array

- Syntax

```
type* variable;
```

type : data type
* : เครื่องหมายที่แสดงว่าตัวแปรที่สร้างเป็นตัวแปรชนิดพอยน์เตอร์
variable : variable name

Dynamic Array

- จองเนื้อที่ในหน่วยความจำสำหรับเก็บข้อมูลในอาร์เรย์ที่ตัวแปรพอยน์เตอร์ชี้ไป
- การเก็บข้อมูลลงไปในอาร์เรย์ต้องจองเนื้อที่ตามขนาดที่ต้องการก่อนโดยฟังก์ชัน malloc ซึ่งมีรูปแบบดังนี้

```
variable = (type *) malloc(size_of_array*sizeof(type));
```

type : ชนิดข้อมูลที่ต้องการจองเนื้อที่

size_of_array :จำนวนอาร์เรย์ที่ต้องการ

sizeof เป็นฟังก์ชันเพื่อหาขนาดของชนิดข้อมูลที่ต้องการ เมื่อนำจำนวนของอาร์เรย์คูณกับขนาดของชนิดข้อมูลทำให้ได้ขนาดทั้งหมดเพื่อจองเนื้อที่ในหน่วยความจำ

Dynamic Array

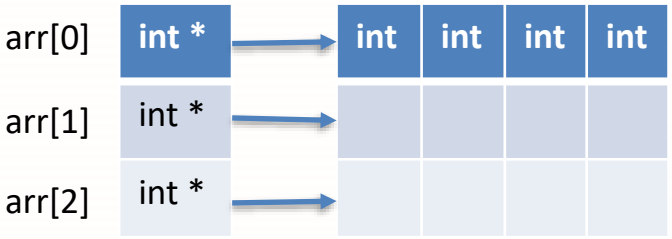
```c
#include<stdio.h>
void main()
{
    int* score;
    int i;
    int number;

     printf("Enter number of students  :");
     scanf("%d",&number);

     score = (int *) malloc(number*sizeof(int));
     printf("\n\nEnter score: \n");

     for  (i=0;i<number;i++)
     {
          scanf("%d", &score[i]);
     }

      printf("\n\n----------------------\n");

     for  (i=0;i<number;i++)
     {
          printf("%d ",score[i]);
     }
}
```

```
Enter number of students  :3


Enter score:
21
22
23


----------------------
21 22 23
```

2-D dynamic array

- Static allocation: array of arrays
  - int arr[3][4]

- Dynamic allocation




```
for(i=0 to 2)
        arr[i]= malloc(sizeof(int)*4)
```

int **arr = malloc(sizeof(int *) * 3)

Realloc()

- Resize the size of a Memory
- Increase or decrease the size of an allocated memory block

- Syntax
  - Pointer = realloc(Pointer, new_size)

- Example

```
int *p;
p = (int*)malloc(5*sizeof(int));
p = realloc(p, 10*sizeof(int));
```

Free()

- Used to de allocate the allocated memory using **malloc** and **calloc**
- Important to release the memory that is not use because it can be used in future

- Syntax
  - Free(Pointer)

- Example

  int *p;

  p = (int*)malloc(5*sizeof(int));

  free(p)

```c
#include<stdio.h>
#include<stdlib.h>

void main()
{
    int *ptr;

    ptr = (int *)malloc(5*sizeof(int));

    *ptr = 10;
    printf("%d",*ptr);

    free(ptr);
}
```