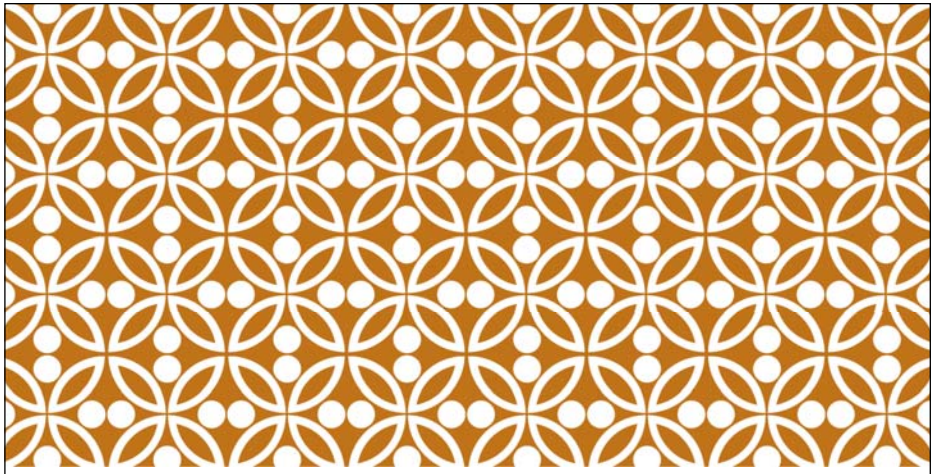


# POINTER

Natsuda Kaothanthong
CS102 (14 October, 2014)



## REVIEW - FUNCTION

- Function
  - ประเภทของฟังชั่น
  - การเรียกใช้งานฟังชั่น
    - Caller
    - Callee
  - การจัดวางฟังชั่นที่สร้างขึ้น
  - Variable
    - Local Variable
    - Global Variable



## ส่วนประกอบของฟังชั่นย่อยที่สร้างขึ้น

```
ชนิดฟังชั่น  ชื่อฟังชั่น(ชนิดของข้อมูลที่ได้รับ  ชื่อข้อมูล)
{
        คำสั่งภายในฟังชั่น

        return   ค่าส่งกลับ;
}
```



## ตัวอย่างฟังชั่น

```
int  computeChange(int price, int pay)
{
    int change;
    change = pay - price;
    return change;
}
```

```
int main()
{
    int total = 250, money = 500, left_over;
    left_over = computechange(total,money);
    printf("You get %d change",left_over);
    return 0;
}
```

# VARIABLE SCOPE

```
float radius;

float computeArea(float r)
{
      float area;
      area = 3.141*r*r;
      radius = radius+2;
      return area;
}

float computePerimeter()
{
      float perimeter;
      perimeter = 2.0*3.141*radius;
      return perimeter;
}
```

```
int main()
{
      radius = 3.0;
      printf("Area of %f radius:
%f\n",radius,computeArea(radius);
      printf("Perimeter of %f radius: %f\n",radius,
computePerimeter(radius);
      return 0;
}
```



# THIS LECTURE

- Pointer
  - ที่อยู่ภายในหน่วยความจำ Memory Address
  - ชนิดข้อมูลแบบพื้นฐาน Primitive Variable
  - ชนิดข้อมูลแบบพอยท์เตอร์ Pointer Variable
- Function
  - การเรียกฟังชั่นโดยใช้ตัวแปลชนิดพอยท์เตอร์ Call by reference



# MEMORY ADDRESS

| ที่อยู่ | Memory |
|---|---|
| 18827 | |
| 18828 | |
| 18829 | |
| 18830 | |
| 18831 | |
| 18832 | |
| 18833 | |
| 18834 | |
| 18835 | |
| 18836 | |
| 18837 | |
| 18838 | |



# PRIMITIVE TYPE VARIABLE

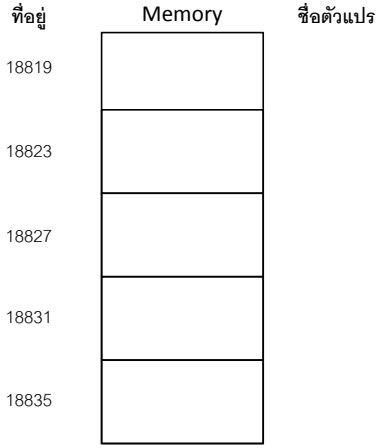
```
int num = 2;

printf("Value is %d",num);

printf("Address is %p",&num);
```

| ที่อยู่ | Memory | ชื่อตัวแปร |
|---|---|---|
| 18827 | | |
| 18828 | | |
| 18829 | | |
| 18830 | | |
| 18831 | | |
| 18832 | | |
| 18833 | | |
| 18834 | | |
| 18835 | | |
| 18836 | | |
| 18837 | | |
| 18838 | | |

# PRIMITIVE TYPE VARIABLE

float num2;

scanf(“%f”, &num2);



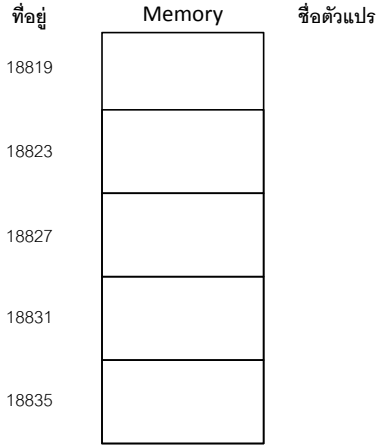



# ตัวอย่าง PRIMITIVE TYPE

int x = 2;

int k = x;

x = 6;






# POINTER VARIABLE

- Pointer variableเป็นตัวแปรชนิดหนึ่ง
  - สำหรับอ้างอิง ตัวแปรที่เชื่อมกับ Pointer variable

ตัวอย่างการประกาศตัวแปรPointer

```
int num = 2;
int *numPtr = &num;
```

```
int x;
int *xPtr;
xPtr = &x;
```

Pointer variableเก็บที่อยู่ของตัวแปรที่ต้องการอ้างอิง

```
printf("Value of xPtr: %p\n",xPtr);

printf("Address of x: %p\n",&x);
```



# POINTER VARIABLE-DEREFERENCING

การกำหนดที่อยู่ในหน่วยความจำให้กับตัวแปรPointer เช่น

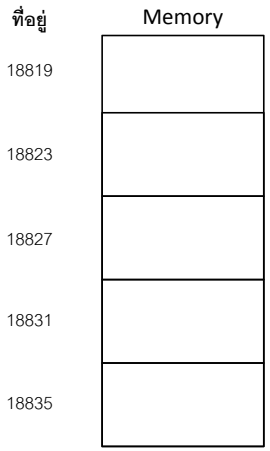
```
int x = 3;
int *xPtr;

xPtr = &x;
```

ตัวแปร x และตัวแปร xPtr ที่เป็น Pointer จะมีความเชื่อมโยงกัน

การดึงค่าของตัวแปร x ผ่านทางตัวแปร xPtr ที่เป็น pointer ทำได้โดยการ Dereferencing โดยใส่เครื่องหมาย * หน้าตัวแปร pointer

```
printf("Value of *xPtr: %d\n",*xPtr);

printf("Value of x: %d\n",x);
```

# POINTER VARIABLE




**ตัวอย่าง**

```
int num = 2;

int *numPtr = &num;

printf("*numPtr : %d", *numPtr);

printf("numPtr : %p", numPtr);

printf("&numPtr : %p", &numPtr);
```

# POINTER VARIABLE

เมื่อต้องการเปลี่ยนค่าของตัวแปรที่เชื่อมกัน ทำได้ 2 วิธี

```
int x = 3;
int *xPtr;

xPtr = &x;
```

1. เปลี่ยนค่าผ่านตัวแปรโดยตรง
   - x = 30;
2. โดยการDereferencing คือการเปลี่ยนค่าผ่านตัวแปรที่เป็น pointer (เครื่องหมาย *)
   - *xPtr = 40;



# POINTER VARIABLE DECLARATION

```
int a;
int *a1Ptr = &a;
```

```
int b;
int *bPtr;
bPtr = &b;
```

```
int d;
int *dPtr = &d;
scanf("%d", dPtr);
```

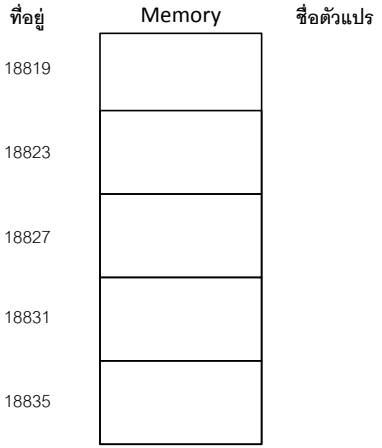
```
int c;
int *cPtr;
*cPtr = &c;
```

```
int e;
int *ePtr = &e;
scanf("%d", &ePtr);
```



# ตัวอย่าง POINTER VARIABLE

```
int x = 2;
int *k = &x;
x = 6;
printf("Value of x: %d\n",x);

*k = 2*3+7/2;
printf("Value of x: %d\n",x);
```



```
1.  int main()
2.  {
3.     int x = 2;
4.     int *xPtr;
5.     xPtr = &x;

6.     printf("Address of x is %p\n",&x);
7.     printf("Value inside memory of xPtr is %p\n",xPtr);
8.     printf("Value of *xPtr is %d\n",*xPtr);

9.     printf("\n----------\n");
10.    x = 35;
11.    printf("New value of x is %d\n",x);
12.    printf("Current value of *xPtr is %d\n",*xPtr);

13.    printf("\n----------\n");
14.    *xPtr = 45;
15.    printf("Current value of x is %d\n",x);
16.    printf("New value of *xPtr is %d\n",*xPtr);
17.    return 0;
18. }
```



# ข้อควรระวังเมื่อประกาศตัวแปรที่เป็น POINTER

1. ชนิดของตัวแปร pointer ต้องมีชนิดเดียวกับตัวแปรที่อ้างอิง

```
float y = 3.141;
int *yPtr = &y;

printf("Value of yPtr: %p\n",yPtr);
printf("Value of *yPtr: %f\n",*yPtr);
```

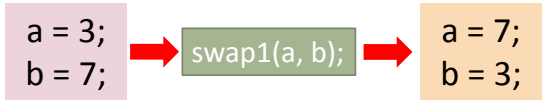
2. หากไม่ต้องการให้ตัวแปร pointer อ้างอิงถึงตัวแปรใด ควรให้ตัวแปรนั้นมีค่า NULL

```
int *kPtr = NULL;

float *jPtr;
jPtr = NULL;

double *zPtr;
*zPtr = NULL;
```



# ทำไมต้องใช้POINTER

- ตัวอย่างฟังชั่น swap1(int a, int b) ทำหน้ากำหนดค่าที่อยู่ในตัวแปร a ให้เป็นค่าที่อยู่ในตัวแปร b




```
void swap1(int a, int b)
{
   int tmp;
   tmp = a;
   a = b;
   b = tmp;
}
```

```
int main()
{
   int x = 1, y = 2;
   swap1(x, y);
   printf("%d %d\n", x, y);
   return 0;
}
```

Call by Value



# FUNCTION-CALL BY REFERENCE

- การส่งผ่านค่า Address (ที่อยู่) ของข้อมูลไปยังฟังชั่น

**ฟังชั่นย่อย**

```
ชนิดฟังชั่น ชื่อฟังชั่น(ชนิดของข้อมูลpointerที่ได้รับ ชื่อข้อมูล)
{
        คำสั่งภายในฟังชั่น

        return ค่าส่งกลับ;
}
```

| ชื่อฟังชั่น(&ชื่อข้อมูล) | **เมื่อต้องการเรียกฟังชั่นย่อย** |
|---|---|

## ตัวอย่าง

```
void addValue(int *x, int *y)
{
    *x = *x + 3;
    *y = *y + 4;
}
int main()
{
   int a = 1, b = 2;
   printf("Before : %d %d\n", a, b);
   addValue(&a, &b);
   printf("After : %d %d\n", a, b);
   return 0;
}
```



## การใช้ POINTER ในฟังชั่น SWAP

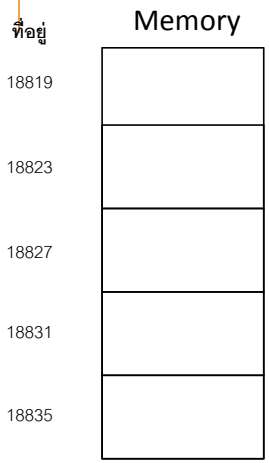
a = 3;
b = 7;
→ swap1(a, b); →
a = 7;
b = 3;

```
void swap1(int *a, int *b)
{
   int tmp;
   tmp = a;
   a = b;
   b = tmp;
}
```

```
int main()
{
   int x = 1, y = 2;
   printf("Before:%d %d\n", x, y);
   swap1(&x, &y);
   printf("%d %d\n", x, y);
   return 0;
}
```



## ทบทวน POINTER




```
int i = 4;
int *iPtr = &i;
```

ข้อใดต่อไปนี้ได้ผลลัพธ์เท่ากัน

1. printf("%p",&iPtr);
2. printf("%p",iPtr);
3. printf("%p",&i);
4. printf("%d",i);
5. printf("%d",*iPtr);



## ทบทวน POINTER

- การกำหนดค่าให้ตัวแปรด้านล่างนี้ถูกต้องหรือไม่

กำหนดตัวแปรดังต่อไปนี้

```
int x = 3;
int *xPtr;
char b = 'Q';
char *bPtr
float c = 21.75;
float *cPtr;
```

1. *xPtr = &x;
2. bPtr = &b;
3. cPtr = &x;

# ทบทวน POINTER

การประกาศตัวแปรพอยท์เตอร์ดังนี้ผิดหรือไม่

1

```
int y = 4;
int *a, *b, *c;
a = &y;
b = &y;
c = &y;
```

2

```
int k = 8;
int *j = &k;
scanf("%d",k);
printf("k is %d",k);
```

# POINTER AND ARRAY

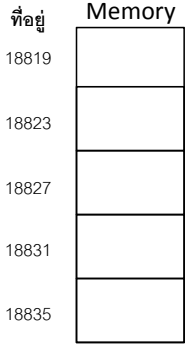


# การประกาศตัวแปร ARRAY

ต้องมี**การกำหนดขนาดของอะเรย์**

```
int a[] = {10, 20, 30, 40, 50};

int a[5] = {10, 20, 30, 40, 50};

int a[3];
a[0] = 10;   a[1] = 20;      a[2] = 30;

int a[];
```

# การใช้ *(ชื่อตัวแปร + INDEX)




- ชื่อตัวแปร Array เป็นการใช้แทนกลุ่มข้อมูล

```
int num[4];
num[0] = 20;
```

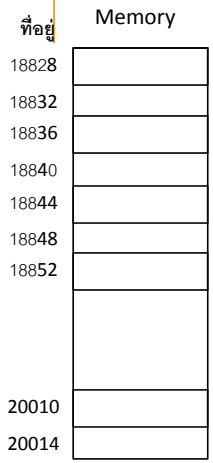
- numเป็นตัวแปรแบบPointer
  - *(**ชื่อตัวแปรArray** + index)
- ใช้เหมือนกับ **ชื่อตัวแปร**[index]

| | |
|---|---|
| num[2] = 8; | *(num+2) = 8; |
| scanf("%d", &num[1]); | scanf("%d", num+1); |

## การใช้ ชื่อตัวแปร[INDEX] และ *(ชื่อตัวแปร+INDEX)

```
char *name = "Hello";
int i;
for(i = 0; i<strlen(name); i++)
{
    printf("%c\t",name[i]);
    printf("%c\n",*(name+i));
}
```

## การใช้POINTERในการชี้ค่าแต่ละค่าในARRAY




```
int *p;
int num[7] = {15,25,35,45,55,65,75};
p = &num;
int i;
for(i = 0; i<7; i++)
{
    printf("%d: %d\n",i,*p);
    p++;
}
printf("---------------\n");
p = &num;
printf("*p: %d\n",*p);
```

## การคัดลอกระหว่าง STRING

char[6] name1 = “Anne”;

char[6] name2 = “Amy”;

name1 = name2;

char *name3 = "Hello";

char *name4 = name3;