



เนื้อหาที่จะเรียนในวันนี้

1. ทบทวน Structure
2. ทบทวน Pointer
3. Pointer to Array
4. Pointer to Structure
5. การใช้งาน pointer เมื่อเขียนโปรแกรม และการ new

บทที่ 2

6. Recursive และการทำงานของ recursive
7. คณิตศาสตร์พื้นฐาน
8. Big O

Structure

1. Definition structure
2. ประกาศตัวแปร
3. การใส่ค่าลงในตัวแปร
 - Assign ค่าด้วยเครื่องหมาย =
 - รับข้อมูลจาก keyboard
 - ใส่ค่าตอนประกาศตัวแปร
4. การดึงค่ามาใช้



3. Structure

type ที่ user สร้างเอง โดยประกอบข้อมูลขึ้น type

Structure is a user-defined datatype which allows us to combine data of different types together.

```
struct student
```

```
{    int Id;  
    float Grade;  
    char Gender;  
};
```

```
struct student std1; 9 bytes
```

std1

int	Id:	
float	Grade:	
char	Gender:	

การใช้งาน

1. Definition structure ✓
2. ประกาศตัวแปร ✓
3. การใส่ค่าลงในตัวแปร
4. การดึงค่ามาใช้



```
struct student  
{  int Id;  
    float Grade;  
    char Gender;  
}std2;
```

std2

int	Id:	
float	Grade:	
char	Gender:	



3.1 การ defined structure และประกาศตัวแปร

```
struct employee  
{   int Id;  
    string Name; หมายถึงเก็บข้อมูลใน C++  
    float Salary;  
};
```

```
struct employee member1, member2;
```

member1

int	Id:	
string	Name:	
float	Salary:	

member2

int	Id:	
string	Name:	
float	Salary:	



3.2 การใส่ค่าลงในตัวแปร structure member1

```
member1.Id=100;  
member1.Name= "Somchai";  
member1.Salary=40000;
```

```
cin >> member2.Id;  
cin >> member2.Name;  
cin >> member2.Salary;
```

101
Jane
41000

int	Id:	100
char[]	Name:	Somchai
float	Salary:	40000

member2

int	Id:	101
char[]	Name:	Jane
float	Salary:	41000



3.3 structure initialization(ใส่ค่าเริ่มต้น)

init = initialization (ใส่ค่าเริ่มต้น)

```
struct structname variable = { val1, val2, ... };
```

member3

int	Id:	102
string	Name:	Kuer
float	Salary:	50000

cin ใส่ค่าของประเภทที่เราใช้

```
struct employee member3 = { 102, "Kuer", 50000};
```



4. Pointer

1. Declaration of a pointer

```
int x, *ptr;
```

4 byte

x



1024

1024 - 1028

4 byte

ptr



1080

2. Pointer Operator

pointer to integer

เก็บค่าแอดเดรส ถ้าแอดเดรสนั้นชี้ integer อยู่

& Address Operator

* Indirection Operator

```
int x, y, *ptr;
```

```
x=5;
```

```
ptr = &x;
```

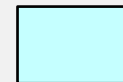
```
y = *ptr;
```

x



1024

y



1050

ptr



1080

&x = Address = 1024

&y = Address = 1050

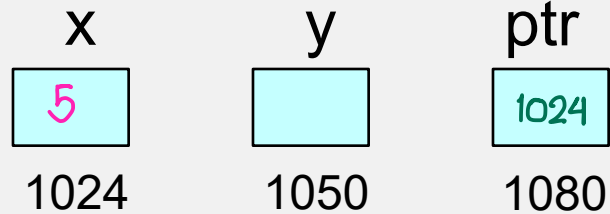
&ptr = Address = 1080

*ptr : 5

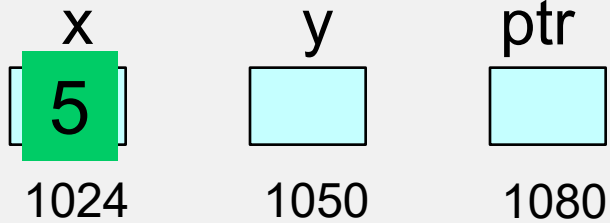


ตัวอย่าง 4.1

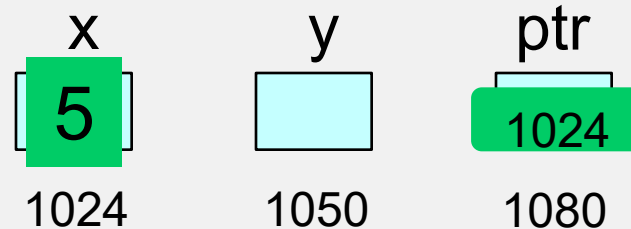
1. `int x, y, *ptr;`



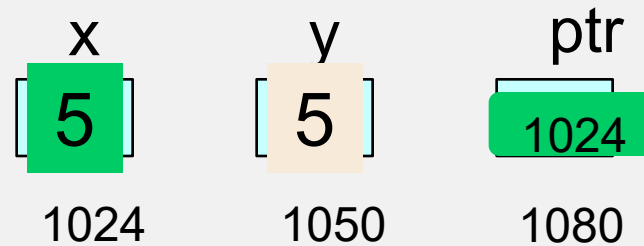
2. `x=5;`



3. `ptr = &x;`

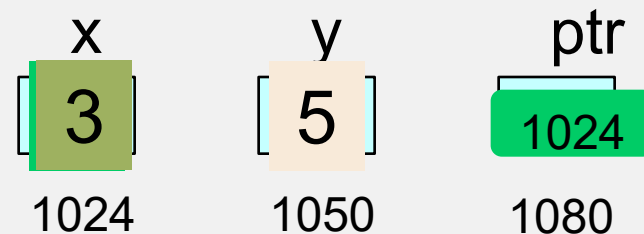


4. `y = *ptr;`



5. `*ptr = 3;`

defined new value

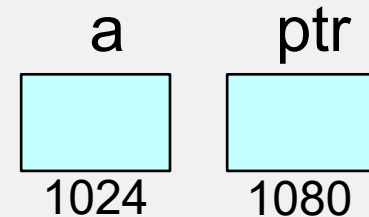




Exercise 1

- ประกาศตัวแปร ชื่อ a มี type float และประกาศตัวแปรชื่อ ptr มี type pointer to float กำหนดให้ a มีค่า 3.14

```
float a = 3.14 , *ptr ;
```



- กำหนดให้ ptr เก็บตำแหน่งของ a

```
ptr = &a ;
```

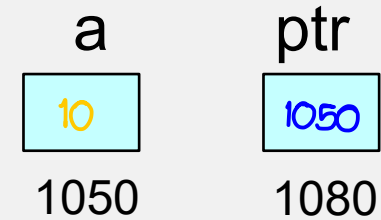
- พิมพ์ค่า 3.14 โดยผ่านทาง ptr

```
cout << *ptr ;
```



ตัวอย่าง 4.2 ทดลองพิมพ์ค่า จากตัวแปร pointer

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {   int a=10 ,*ptr;
5.     ptr = &a;
6.     cout << "*ptr      = " << *ptr << endl;
7.     cout << "a      = " << a << endl;
8. }
```





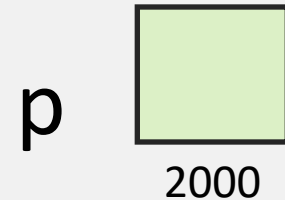
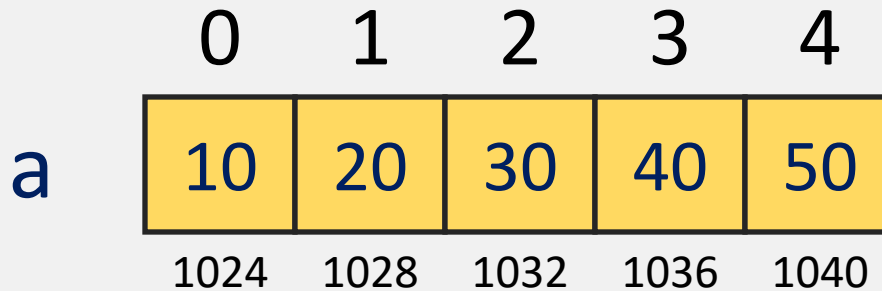
4.2 Pointer advanced

4.2.1 Pointer and array

ตัวเก็บข้อมูลของ Array

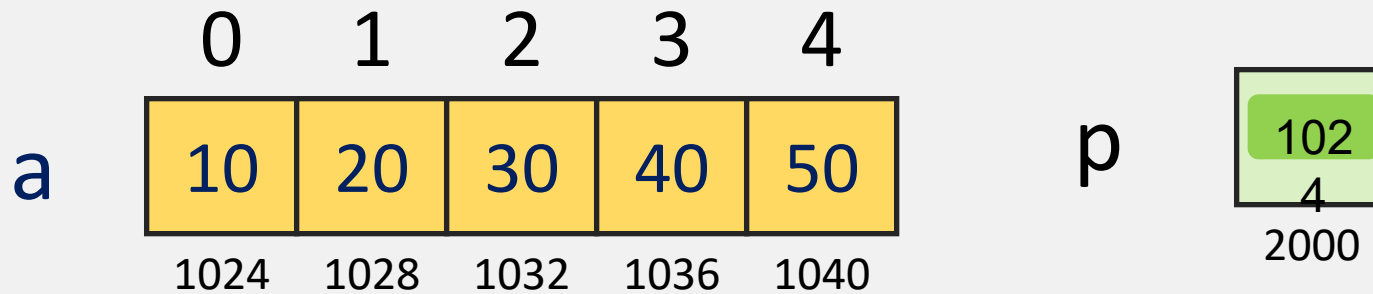
```
int a[5]={10,20,30,40,50};
```

```
int *p;
```



× 5
2000

×: 5



a = 1024

a+1 = 1028

a[0] = 10

a[1] = 20

***a** = 10

***(a+1)** = 20

***a+1** = 11

&a[0] = 1024

a++

p = a (1024)

p++ = 1028

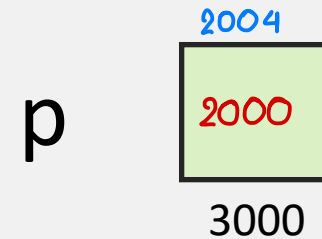
***p** = 20

***(p++)** = 30 (ต่อของเก่า)

ห้ามขยับ array



	0	1	2	3	4
a	5.2	3.6	12	4.5	2.8
	2000	2004	2008	2012	2016



```
float a[5]={5.2,3.6,12,4.5,2.8};
```

```
float *p=a;
```

a	=	2000	*a	=	5.2
a+2	=	2008	*(a+2)	=	12
a[0]	=	5.2	*a+1	=	6.2
a[3]	=	4.5	*(a++)	=	ไม่ได้ error
			&a[3]	=	2012

p	=	a	=	&a
p++	=	2004		
*p	=	3.6		
*(p++)	=	12	(ต่อ	
ของเก่า)				
เป็น pointer				

ตัวอย่าง 9.3 ความแตกต่างระหว่าง Array กับ Pointer to array

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{    char array[10];
```

```
    char *ptr;
```

```
    array="hello"; /*wrong */
```

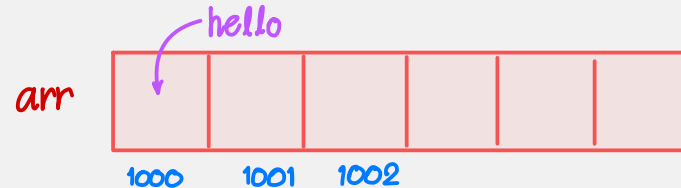
```
    ptr = "hello";
```

```
    cout << ptr;
```

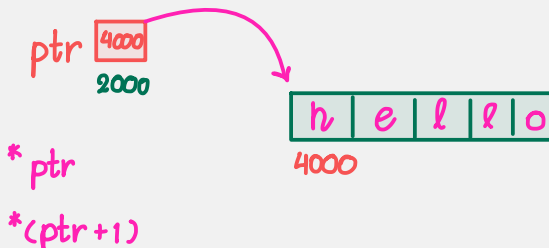
```
}
```

array = "hello";

strcpy (array, "hello");



anonymus array



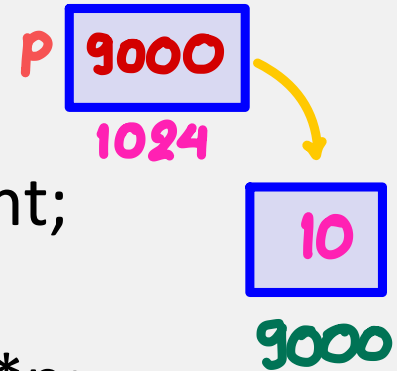


```
#include <iostream>
using namespace std;
int main()
{
    int *p ,a;
    a=10;
    p=&a;
    cout << *p;
}
```

p 1080
1024

a 10
1080

```
#include <iostream>
using namespace std;
int main()
{
    int *p;
    p=new int;
    *p=10;
    cout << *p;
}
```





4.3.2 Pointer to Structure

struct record

{ **4byte** int id;

4byte float grade;

};

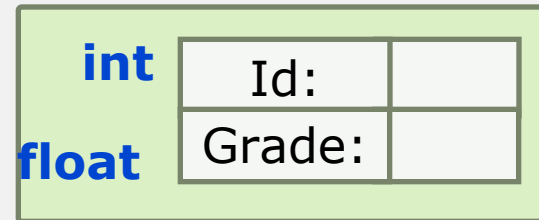
struct record std1, std2;

struct record *ptr;

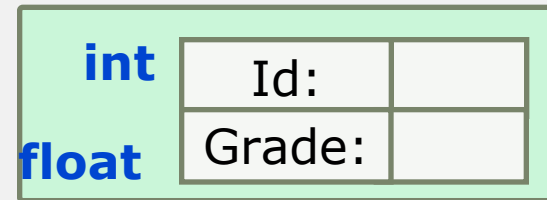
ptr

3018
3040

2000

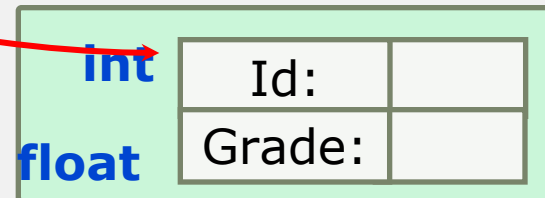


std1



3018

std2



3040



```
struct record std1, std2;  
struct record *ptr;
```

```
std1.Id=100;
```

```
std1.Grade=2.75;
```

```
ptr=&std2;
```

```
(*ptr).Id=101; หรือ
```

```
ptr->Id=101;
```

```
ptr ->Grade=3.18;
```

std1

int	Id:	100
float	Grade:	2.75

3018

ptr

2000

std2

int	Id:	101
float	Grade:	3.18

3040

} ใส่ค่าผ่านตัวแปร

} ใส่ค่าผ่าน pointer

ptr → Id = 101;
ptr → Grade = 3.18;



```
#include <iostream>
using namespace std;
struct record
{ int id; data
  string name;
};
int main()
{ struct record data, *p;
  data.id=100;
  data.name="Somchai";
  p=&data;
  cout << p->id <<
    " " << p->name << endl;
}
```

Diagram illustrating memory addresses for the first example:

- Variable `data` (struct record) is located at memory address 4000 and contains `id=100` and `name="Somchai"`.
- Variable `p` (pointer to struct record) is located at memory address 2000 and contains the address 4000, pointing to the `data` variable.

```
#include <iostream>
using namespace std;
struct record
{ int id;
  string name;
};
int main()
{ struct record *p;
  p = new struct record;
  p->id =200;
  p->name="Teetus";
  cout << p->id << " "
    << p->name << endl;
}
```

Diagram illustrating memory addresses for the second example:

- Variable `p` (pointer to struct record) is located at memory address 1000 and contains the address 5000, pointing to the dynamically allocated struct record.
- The dynamically allocated struct record is located at memory address 5000 and contains `id=200` and `name="teetus"`.



```
#include <iostream>
using namespace std;
struct record
{   int value;
    struct record *next;
};
```

```
int main()
{   struct record *n1;
    n1=new struct record;
    n1->value=10;
    n1->next=NULL;
    cout << n1->value <<endl;
}
```

