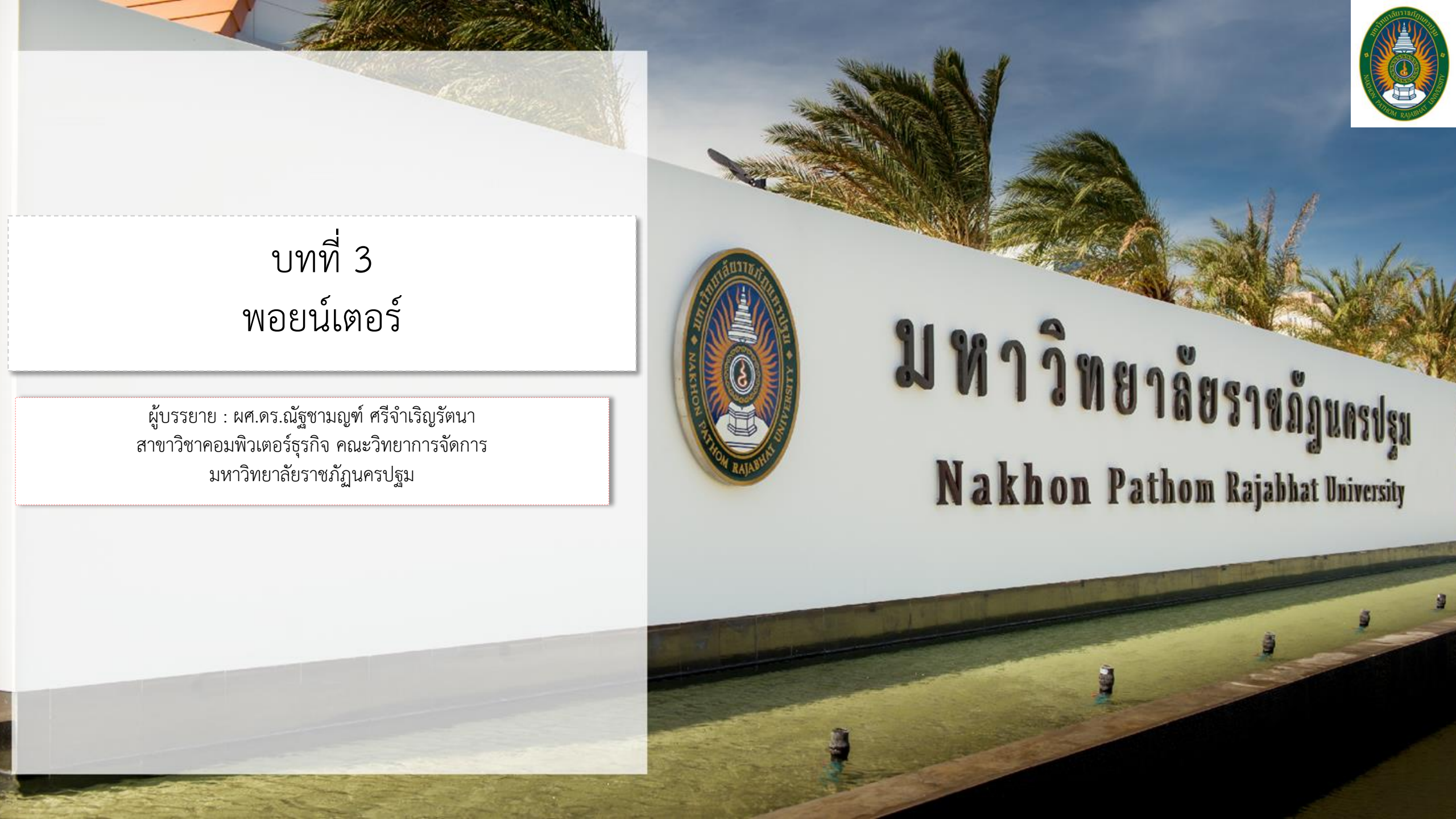
มหาวิทยาลัยราชภัฏนครปฐม



บทที่ 3
พอยน์เตอร์

ผู้บรรยาย : ผศ.ดร.ณัฐชามญฑ์ ศรีจำเริญรัตนา
สาขาวิชาคอมพิวเตอร์ธุรกิจ คณะวิทยาการจัดการ
มหาวิทยาลัยราชภัฏนครปฐม

พอยน์เตอร์

ตัวแปรชนิดพอยน์เตอร์ สามารถระบุตำแหน่งของตัวแปรที่อยู่ในหน่วยความจำเพื่อนำมาใช้งานได้

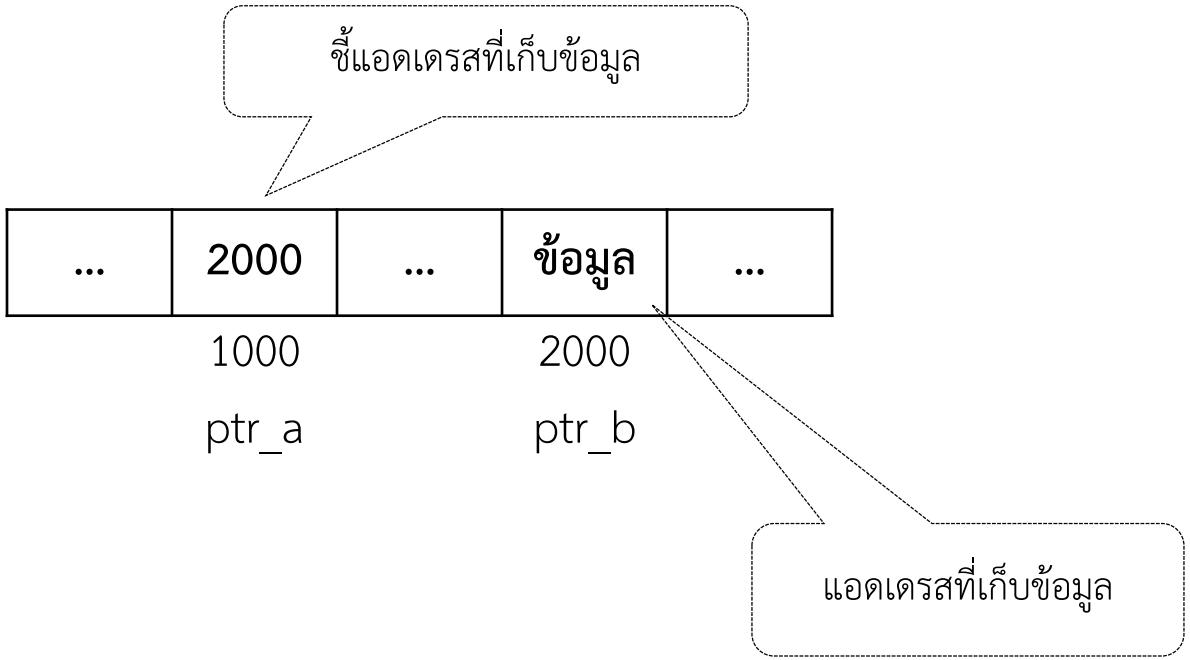
โดยปกติแล้ว

- ตัวแปรจะมีพื้นที่ในหน่วยความจำแต่ละตำแหน่งซึ่งเรียกว่า แอดเดรส
- ข้อมูลแอดเดรสเป็นตัวแปรที่เก็บข้อมูลไว้ในตัวแปรชนิดพอยน์เตอร์
- ตัวแปรชนิดพอยน์เตอร์จะทำการชี้ไปยังตำแหน่งที่เก็บข้อมูล

สำหรับภาษาซี คุณสมบัติของพอยน์เตอร์จะทำให้ผู้ใช้สามารถทำงานได้ง่ายขึ้น และยังช่วยให้ผู้เขียนโปรแกรมสามารถทำงานได้มีความยืดหยุ่น แม้ว่าพอยน์เตอร์จะมีรูปแบบการใช้งานที่ซับซ้อน แต่ถ้าทำความเข้าใจรูปแบบการทำงาน พอยน์เตอร์ก็เป็นตัวแปรหนึ่งที่ช่วยให้การทำงานได้ง่ายขึ้น

3.1 แนะนำพอยน์เตอร์

โครงสร้างข้อมูลแบบพอยน์เตอร์

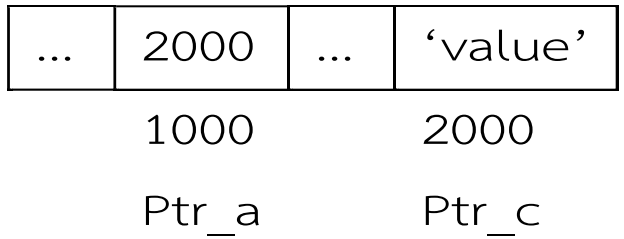


3.1 แนะนำพอยน์เตอร์(ต่อ)

ประเภทของตัวแปรชนิดพอยน์เตอร์

การประกาศใช้ตัวแปรประเภทพอยน์เตอร์ ต้องบอกด้วยว่าเป็นพอยน์เตอร์ชี้ไปยังตัวแปรชนิดใด โดยสามารถแบ่งได้ออกเป็น 2 ประเภท ดังนี้

Direct Pointer variable หมายถึง ตัวแปรชนิดพอยน์เตอร์ที่เก็บตำแหน่งของข้อมูลภายในหน่วยความจำโดยตรง



การเก็บข้อมูลแบบ Direct Pointer variable

3.1 แนะนำพอยน์เตอร์(ต่อ)

Indirect Pointer variable หมายถึง ตัวแปรชนิดพอยน์เตอร์ที่เก็บตำแหน่งของตัวแปรชนิดพอยน์เตอร์อีกตัวหนึ่ง หรือบางครั้งจะเรียกว่า Pointer to Pointer

…	2000	…	3000	…	‘value’
	1000		2000		3000
	Ptr_a		Ptr_b		Ptr_c

การเก็บข้อมูลแบบ Indirect Pointer variable

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์

- ค่าของตัวแปรชนิดพอยน์เตอร์ คือ ค่าที่ใช้อ้างอิงตำแหน่งในหน่วยความจำภายในเครื่องคอมพิวเตอร์
- ดังนั้นข้อมูลที่ตัวแปรชนิดพอยน์เตอร์เก็บจะเป็นตำแหน่งของหน่วยความจำที่ใช้อ้างอิงตำแหน่งจริงที่เก็บข้อมูล
- การประกาศตัวแปรพอยน์เตอร์ต้องระบุชนิดของตัวแปรของข้อมูลที่จะเก็บในตำแหน่งหน่วยความจำที่พอยน์เตอร์ชี้
- ความแตกต่าง
  - ตัวแปรทั่วไปเป็นตัวแปรที่สร้างเพื่อให้เก็บข้อมูลที่เราต้องการ
  - ตัวแปรพอยน์เตอร์เป็นตัวแปรที่ถูกสร้างขึ้นมาเพื่อเก็บค่าตำแหน่งของข้อมูลที่เก็บข้อมูลในหน่วยความจำ
- ตัวแปรชนิดพอยน์เตอร์สามารถแบ่งได้ 2 แบบ คือ
  - 3.2.1 การประกาศตัวแปรแบบพอยน์เตอร์ (*)
  - 3.2.2 การประกาศตัวแปรแบบกำหนดค่าตำแหน่งแอดเดรส (&)

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

3.2.1 การประกาศตัวแปรแบบพอยน์เตอร์ (*)

การประกาศตัวแปรแบบพอยน์เตอร์ ใช้สัญลักษณ์ * เรียกอีกชื่อว่า indirection operator หรือ de-referencing operator เป็นเครื่องหมายใช้เพื่ออ้างอิงข้อมูลที่ตัวแปรพอยน์เตอร์ชี้อยู่ โดยจะต้องประกาศประเภทของตัวแปรพอยน์เตอร์ให้สอดคล้องกับประเภทของตัวแปรที่เราต้องการ

รูปแบบการใช้งาน

```
ประเภทตัวแปร *ชื่อตัวแปร;
```

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.1 การประกาศใช้ตัวแปรแบบพอยน์เตอร์ (*)

```
01  int *pnPtr;
02  double *pdPtr;
03  int *pnPtra, *pnPtrb;
04  int *pnPtra, pnPtrb;
```

ตัวอย่างที่ 3.2 การประกาศใช้ตัวแปรทั่วไป

```
01  int pnPtr;
02  double pdPtr;
03  int pnPtra, pnPtrb;
```

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

3.2.2 การประกาศตัวแปรแบบกำหนดค่าตำแหน่งแอดเดรส (&)

การประกาศตัวแปรแบบกำหนดค่าเริ่มต้น ใช้สัญลักษณ์ & เรียกอีกชื่อว่า address of operator เพื่อคืนค่าตำแหน่งแอดเดรสของตัวแปรชนิดพอยน์เตอร์ โดยการกำหนดค่าชนิดนี้ต้องสอดคล้องกับชนิดข้อมูลของตัวแปรพอยน์เตอร์เท่านั้น

รูปแบบการใช้งาน

```
ชื่อตัวแปรชนิดพอยน์เตอร์ = &ตัวแปรที่เก็บแอดเดรสของข้อมูล;
```

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.4 การแสดงค่าตัวแปรแบบพอยน์เตอร์

```
01  #include <iostream>
02  using namespace std;
03  int main()
04  {
05     int holdvalue;
06     int *getvalue;
07     holdvalue = 8;
08     getvalue = &holdvalue ;
09     *getvalue = 55;
10      cout<<"holdvalue = "<<holdvalue<<endl;
11     cout<<"getvalue = "<<getvalue<<endl;
12     cout<<"*getvalue = "<<*getvalue<<endl;
13     cout<<"&getvalue = "<<&getvalue<<endl;
14     return 0;
15  }
```

รูปแบบการประกาศการใช้ตัวแปรแบบกำหนดค่าเริ่มต้น

ผลการทำงานของโปรแกรม

```
holdvalue = 55
getvalue = 0x22fd7c
*getvalue = 55
&getvalue = 0x22fd70
```

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.5 จงแสดงข้อมูลในตัวแปรแต่ละขั้นตอนการทำงาน

สิ้นสุดการทำงาน ขั้นตอนที่	**ค่าในตัวแปร**			
07-08				
		0x22fd7c		8
	...	0x22fd70 getvalue	...	0x22fd7c holdvalue
09				
		0x22fd7c		55
	...	0x22fd70 getvalue	...	0x22fd7c holdvalue

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.6 การประกาศใช้งานแบบพอยน์เตอร์

```
01  #include <iostream>
02  using namespace std;
03  int   main()
04  {
05     int ptr_data = 15;
06     int *pointer;
07     pointer = &ptr_data;
08     cout<<"ptr_data is “<<ptr_data<<endl;
09     cout<<"Address of ptr_data is "<<&ptr_data<<endl;
10     cout<<"Address of pointer is "<<&pointer<<endl;
11     cout<<"value of pointer is "<<pointer<<endl;
12     return 0;
13  }
```

ผลการทำงานของโปรแกรม

```
ptr_data is 15
Address of ptr_data is 0x22fd7c
Address of pointer is 0x22fd70
value of pointer is 0x22fd7c
```

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

รูปแบบการแสดงค่าในตัวแปร

	0x22fd7c		15	
	0x22fd70		0x22fd7c	
...	pointer	...	ptr_data	

ข้อสังเกต โครงสร้างการประกาศค่าตัวแปรแบบพอยน์เตอร์ สามารถประกาศได้แตกต่างกันได้ดังนี้

Int *VariableName;

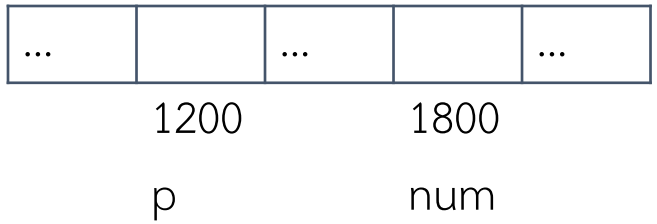
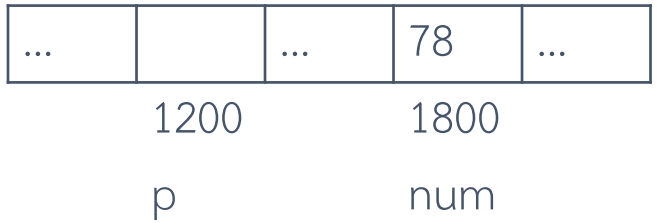
Int* VariableName;

Int *VariableName;

เครื่องหมายดอกจัน(*) สามารถปรากฏตำแหน่งใด ๆ ระหว่างชนิดข้อมูลของตัวแปรและชื่อของตัวแปร

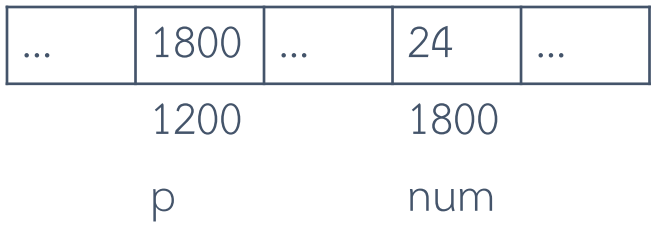
3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.7 จงอธิบายค่าในตัวแปรแอดเดรสแต่ละค่าการทำงาน โดยกำหนดค่าเริ่มต้นให้ ตัวแปร p อยู่ที่แอดเดรสตำแหน่ง 1200 และ ตัวแปร num อยู่ที่แอดเดรสตำแหน่ง 1800

สิ้นสุดการทำงานขั้นตอนที่	ค่าในตัวแปร	คำสั่งที่กำหนด
1-2		ประกาศให้ - ตัวแปร p เป็นตัวแปรพอยน์เตอร์ชนิด int - ตัวแปร num เป็นตัวแปรพอยน์เตอร์ชนิด int
3		กำหนดให้ - ตัวแปร num เก็บค่า 78

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.7 (ต่อ)

สิ้นสุดการทำงาน ขั้นตอนที่	ค่าในตัวแปร	คำสั่งที่กำหนด
4		กำหนดให้ - ตัวแปร p เก็บตำแหน่งแอดเดรสของตัวแปร num ดังนั้น p จึงเก็บค่าแอดเดรส 1800 ของตัวแปร num
5		กำหนดให้ - ตัวแปรชนิดพอยน์เตอร์ *p = 24 แต่ *p ชี้ไปที่แอดเดรส 1800 เมื่อไปที่แอดเดรส 1800 พบตัวแปร num เก็บค่า 24 จึงทำการเปลี่ยนค่าที่ตำแหน่ง 1800 เป็นค่าที่พอยน์เตอร์ p กำหนดเป็น 24

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.8 จงแสดงผลการทำงานของตัวแปรพอยน์เตอร์ ต่อไปนี้

```
01  #include <iostream>
02  using namespace std;
03  int main(){
04      int *p;
05      int num1 = 5;
06      int num2 = 10;
07      cout<<endl<<"declaring variable";
08      cout<<endl<<"*p"<<endl<<"num1 = 5"<<endl<<"num2 = 10";
09      p = &num1;
10      cout<<endl<<".....1. p = &num1.....";
11      cout<<endl<<"address of num1 ="<<&num1;
12      cout<<endl<<"value of num1 ="<<num1;
13      cout<<endl<<"address of p ="<<*p;
14      cout<<endl<<"================";
15      *p=10;
```

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.8 (ต่อ)

```
16    cout<<endl<<".....2. *p=10.....";
17    cout<<endl<<"value of num1 ="<<num1;
18    cout<<endl<<"value of *p ="<<*p;
19    cout<<endl<<"================="<<endl;
20    p=&num2;
21    cout<<endl<<".....3. p=&num2.....";
22    cout<<endl<<"value of &num2 ="<<&num2;
23    cout<<endl<<"value of p ="<<p;
24    cout<<endl<<"================="<<endl;
25    *p=2*(*p);
26    cout<<endl<<".....4. *p=2*(*p).....";
27    cout<<endl<<"value of num2 ="<<num2;
28    cout<<endl<<"value of *p ="<<*p;
29    return 0;
30  }
```

หมายเหตุ ผลการรันโปรแกรมค่าตัวแปร &num1 และ p จะมีค่าที่แตกต่างกันในแต่ละเครื่องที่ใช้รันโปรแกรม

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ผลการทำงานของโปรแกรม

```
declaring variable
*p
num1 = 5
num2 = 10
.......1. p = &num1.......
address of num1 =0x22fd74
value of num1 =5
address of p =5
==================
.......2. *p=10.......
value of num1 =10
value of *p =10
==================
```

```
.......3. p=&num2.......
value of &num2 =0x22fd70
value of p =0x22fd70
==================
.......4. *p=2*(*p).......
value of num2 =20
value of *p =20
```

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.9 จงอธิบายการทำงานแต่ละขั้นตอนของพอยน์เตอร์ โดยใช้ตัวอย่างที่ 3.8 ส่วนโปรแกรมและผลการทำงาน

1. p = &num1

...	0x28ff40	...	5	...	10	...
			0x28ff40			
	p		Num1		Num2	

2. *p=10

...	0x28ff40	...	10	...	10	...
			0x28ff40			
	p		Num1		Num2	

3. p=&num2

...	0x28ff3c	...	10	...	10	...
			0x28ff40		0x28ff3c	
	p		Num1		Num2	

4. *p=2*(*p)

...	0x28ff3c	...	10	...	10	...
			0x28ff40		0x28ff3c	
	p		Num1		Num2	

3.2 การประกาศค่าตัวแปรชนิดพอยน์เตอร์(ต่อ)

ตัวอย่างที่ 3.10 การกำหนดการอ้างอิงแอดเดรส

```
01  #include <iostream>
02  using namespace std;
03  int main()
04  {
05      int data = 44;
06
07      int &refdata = data;
08      cout << data << endl;
09      cout << refdata << endl;
10
11      refdata = 55;
12      cout << refdata << endl;
13      cout << data << endl;
14
15      data = 66;
16      cout << data << endl;
17      cout << refdata << endl;
18      return 0;
19  }
```

ผลการทำงานของโปรแกรม

```
44
44
55
55
66
66
```

3.3 ตัวแปรแบบไดนามิก

- การจองพื้นที่ในหน่วยความจำในระบบและการคืนพื้นที่ในหน่วยความจำให้กับระบบในระหว่างการทำงานของโปรแกรมพอยน์เตอร์
- ตัวแปรที่สร้างขึ้นในระหว่างการทำงานของโปรแกรม เรียกว่าตัวแปรแบบไดนามิก การจัดการพื้นที่ในหน่วยความจำสามารถใช้โอเปอเรเตอร์ new และ delete เพื่อใช้สร้างและคืนพื้นที่ในหน่วยความจำ ตามลำดับ

<u>หมายเหตุ</u> โปรแกรมภาษาซี++ new และ delete เป็นคำสงวน

3.3 ตัวแปรแบบไดนามิก(ต่อ)

3.3.1 ตัวดำเนินการ new

ตัวดำเนินการ new ใช้จอง (allocate) พื้นที่ในหน่วยความจำ การใช้งานโอเปอเรเตอร์ new จำเป็นต้องระบุชนิดของข้อมูลและขนาดของหน่วยความจำ โดยคำสั่งที่ใช้ในการจองหน่วยความจำ คือ คำสั่ง new ซึ่งมี 2 รูปแบบดังนี้

โอเปอเรเตอร์ new แบบที่ 1 คำสั่งที่ใช้ในการจองหน่วยความจำ

```
ตัวแปรพอยน์เตอร์ =  new ชนิดตัวแปร;
```

3.3 ตัวแปรแบบไดนามิก(ต่อ)

ตัวอย่างที่ 3.11 จงแสดงการใช้ตัวดำเนินการ new เพื่อทำการจองพื้นที่ในหน่วยความจำ

```
int *p;
p = new int;
หรือ
int *p=new int;
```

อธิบายการทำงาน

- สร้างตัวแปรชนิดจำนวนเต็มเพื่อเก็บแอดเดรสที่จองในหน่วยความจำ
- โดยมีตัวแปรชนิดพอยน์เตอร์ p ชี้อยู่ที่หน่วยความจำ

3.3 ตัวแปรแบบไดนามิก(ต่อ)

โอเปอเรเตอร์ new แบบที่ 2 สามารถใช้งานจองพื้นที่ในหน่วยความจำและพื้นที่ที่จองนี้จะไม่ทำการแบ่งพื้นที่การใช้งานกับตัวแปรอื่น ๆ และผู้พัฒนาโปรแกรมสามารถกำหนดได้ว่าจะให้คืนพื้นที่ให้กับหน่วยความจำช่วงเวลาใด

รูปแบบคำสั่งที่ใช้ในการจองหน่วยความจำ แบบที่ 2

```
ตัวแปรพอยน์เตอร์ =  new ชนิดตัวแปร[ขนาดของอาร์เรย์];
```

3.3 ตัวแปรแบบไดนามิก(ต่อ)

ตัวอย่างที่ 3.12 การจองพื้นที่ในหน่วยความจำ

```
int *p;
p = new int[10];
```

อธิบายการทำงาน

จองพื้นที่ในหน่วยความจำที่ติดต่อกัน 10 อิลิเมนต์ แต่ละอิลิเมนต์กำหนดให้มีชนิดข้อมูลแบบจำนวนเต็มและมีตัวแปร p ชี้อยู่ที่ตำแหน่งเริ่มต้นของหน่วยความจำที่จองขึ้นมา

3.3 ตัวแปรแบบไดนามิก(ต่อ)

ตัวอย่างที่ 3.13 การใช้งานตัวดำเนินการ new

```
01  #include <iostream>
02  using namespace std;
03  int main()
04  {
05     int listSize;
06     cout << "Enter the array size: ";
07     cin >> listSize;
08     int *list = new int [listSize];
09      for ( int j = 0 ; j < listSize ; j++ )    {
10        cout << "Enter the array elements: [“;
11        cout<<j+1<<"] ";
12        cin >> list[j];
13      }
14      for ( int k = 0 ; k < listSize ; k++ ){
15         cout <<"array element no."<< k+1 ;
16         cout <<"="<<list[k] << endl;
17      }
18     return 0;
19  }
```

ผลการทำงานของโปรแกรม

```
Enter the array size: 5
Enter the array elements: [1] 10
Enter the array elements: [2] 20
Enter the array elements: [3] 30
Enter the array elements: [4] 40
Enter the array elements: [5] 50
array element no.1=10
array element no.2=20
array element no.3=30
array element no.4=40
array element no.5=50
```

3.3 ตัวแปรแบบไดนามิก(ต่อ)

3.3.2 ตัวดำเนินการ delete

ตัวดำเนินการ delete ใช้คืนพื้นที่ที่ถูกตัวแปรพอยน์เตอร์ชี้ใช้งานให้กับหน่วยความจำ

รูปแบบตัวดำเนินการ delete

```
delete[] ตัวแปรพอยน์เตอร์
 หรือ
delete ตัวแปรพอยน์เตอร์
```

3.3 ตัวแปรแบบไดนามิก(ต่อ)

ตัวอย่างที่ 3.14 การใช้งานตัวดำเนินการ delete

```
01  #include <iostream>
02  using namespace std;
03  int main()
04  {
05     int *pt1 =new int;
06     *pt1=100;
07     cout<<"Before delete variable pt1 ="<<*pt1<<endl;
08     delete[] pt1;
09     *pt1=110;
10     cout<<"After delete variable pt1 ="<<*pt1;
11     return 0;
12  }
```

ผลการทำงานของโปรแกรม

```
Before delete variable pt1 =100
After delete variable pt1 =110
```

3.3 ตัวแปรแบบไดนามิก(ต่อ)

ตัวอย่างที่ 3.15 การทำงานพอยน์เตอร์

```
01  #include <iostream>
02  using namespace std;
03  int main()
04  {
05      int *p;
06      int *q;
07      p = new int;
08      *p = 34;
09      cout << "Line 10: p = " << p<< ", *p = " << *p << endl;
10      q = p;
11      cout << "Line 12: q = " << q<< ", *q = " << *q << endl;
12      *q = 45;
13      cout << "Line 14: p = " << p<< ", *p = " << *p << endl;
```

3.3 ตัวแปรแบบไดนามิก(ต่อ)

ตัวอย่างที่ 3.15 (ต่อ)

```
14      cout << "Line 15: q = " << q<< ", *q = " << *q << endl;
15      p = new int;
16     *p = 18;
17      cout << "Line 18: p = " << p<< ", *p = " << *p << endl;
18      cout << "Line 19: q = " << q<< ", *q = " << *q << endl;
19      delete q;
20      q = NULL;
21      q = new int;
22      *q = 62;
23      cout << "Line 24: p = " << p<< ", *p = " << *p << endl;
24      cout << "Line 25: q = " << q<< ", *q = " << *q << endl;
25      return 0;
26  }
```

3.3 ตัวแปรแบบไดนามิก(ต่อ)

ผลการทำงานของโปรแกรม

```
Line 10: p = 0x987cf0, *p = 34
Line 12: q = 0x987cf0, *q = 34
Line 14: p = 0x987cf0, *p = 45
Line 15: q = 0x987cf0, *q = 45
Line 18: p = 0x987d10, *p = 18
Line 19: q = 0x987cf0, *q = 45
Line 24: p = 0x987d10, *p = 18
Line 25: q = 0x987cf0, *q = 62
```

3.4 ตัวแปรพอยน์เตอร์กับตัวแปรอาร์เรย์

- ตัวแปรอาร์เรย์ (array) หรือตัวแปรชุด เป็นตัวแปรที่มีการจองพื้นที่ในหน่วยความจำเพื่อให้สามารถเก็บข้อมูลได้มากกว่า 1 ค่า
- แต่ข้อมูลจะต้องเป็นข้อมูลชนิดเดียวกันและมีขนาดคงที่
- ข้อมูลชนิดอาร์เรย์คล้ายกับข้อมูลชนิดพอยน์เตอร์เนื่องจากใช้เก็บค่าแอดเดรสเช่นเดียวกัน
- วิธีการอ้างข้อมูลและตำแหน่งที่อยู่ในหน่วยความจำสามารถทำได้ดังนี้

รูปแบบที่ 1 การอ้างอิงแอดเดรสของอาร์เรย์

ตัวแปรอาร์เรย์+ลำดับ หรือ &ตัวแปรอาร์เรย์[ลำดับ]

รูปแบบที่ 2 การอ้างอิงข้อมูลภายในอาร์เรย์

ตัวแปรอาร์เรย์[ลำดับ] หรือ *(ตัวแปรอาร์เรย์ + ลำดับ)

3.4 ตัวแปรพอยน์เตอร์กับตัวแปรอาร์เรย์(ต่อ)

- รูปแบบคำสั่งที่ใช้เพื่อจองหน่วยความจำโดยใช้ตัวแปรพอยน์เตอร์ทำการชี้ตำแหน่งเริ่มต้นของหน่วยความจำที่จองและใช้พอยน์เตอร์เหมือนกับอาร์เรย์
- พอยน์เตอร์มีจุดเด่นที่ทำงานได้รวดเร็ว ไม่เปลืองเนื้อที่ในหน่วยความจำ
- การใช้งานพอนย์เตอร์นั้นมีจุดประสงค์เพื่อเก็บค่าที่อยู่ของข้อมูล
- แต่การดูตำแหน่งที่อยู่ของข้อมูลที่ถูกเก็บอยู่นั้น สามารถทำได้สองวิธี คือ
  - ผ่านตัวแปรพอยน์เตอร์
  - ผ่านสัญลักษณ์ของ memory address

3.4 ตัวแปรพอยน์เตอร์กับตัวแปรอาร์เรย์(ต่อ)

ตัวอย่างที่ 3.16 จงเขียนโปแกรมแสดงค่าของตัวแปรพอยน์เตอร์กับอาร์เรย์

```
01  #include <iostream.h>
02  #include <conio.h>
03  int main()
04  {
05     int var_a[5] = {2,4,8,16,32};
06     int *Ptr_b;
07     Ptr_b = var_a;
08     cout << "Ptr_b[0] = " << Ptr_b[0] << endl;
09     cout << "Ptr_b[1] = " << Ptr_b[1] << endl;
10     Ptr_b = &var_a[2];
11     Ptr_b[1] = 10;
12     cout << "var_a[1] = " << var_a[1] << endl;
13     cout << "var_a[3] = " << var_a[3] << endl;
14     getch();
15     return 0;
16  }
```

ผลการทำงานของโปรแกรม

```
Ptr_b[0] = 2
Ptr_b[1] = 4
var_a[1] = 4
var_a[3] = 10
```

3.4 ตัวแปรพอยน์เตอร์กับตัวแปรอาร์เรย์(ต่อ)

ตัวอย่างที่ 3.17 การแสดงค่าพอยน์เตอร์กับอาร์เรย์อีกแบบ

```
01  #include <iostream>
02  using namespace std;
03  int main()
04  {
05          int var_data[5] = {0,1,2,3,4};
06          int *ptr;
07          ptr = var_data;
08          cout<<var_data[0]<< endl ;
09          cout<<*ptr<< endl ;
10          cout<<*(ptr+1)<< endl ;
11          cout<<*(var_data+2)<< endl ;
12          cout<<ptr[3]<< endl ;
13          return 0;
14  }
```

ผลการทำงานของโปรแกรม

```
0
0
1
2
3
```

มหาวิทยาลัยราชภัฏนครปฐม