C프로그래밍

Lecture 12. 포인터의 이해

동덕여자대학교
데이터사이언스 전공
권 범

목차

01. 포인터란 무엇인가?

02. 포인터와 관련 있는 & 연산자와 * 연산자

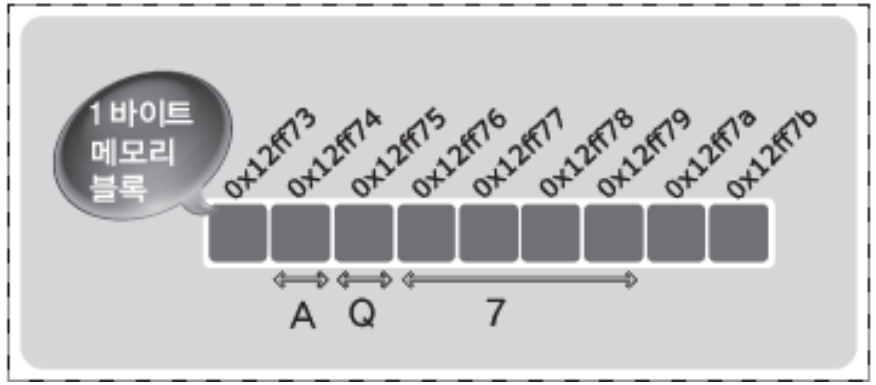
03. 연습 문제

01. 포인터란 무엇인가?

❖ 주소 값의 저장을 목적으로 선언되는 포인터 변수

- 변수 num이 저장되기 시작한 주소 0x12ff76이 변수 num의 주소 값입니다.
- 이러한 정수 형태의 주소 값을 저장하는 목적으로 선언되는 것이 포인터 변수입니다.

```
int main(void)
{
     char ch1 = 'A';
     char ch2 = 'Q';
     int num = 7;
     ...
}
```



포인터란 메모리의 주소 값을 저장하기 위한 변수입니다.

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

01. 포인터란 무엇인가?

❖ 포인터 변수 선언하기

- 포인터 변수에도 타입(Type, 형)이 존재합니다.
- 가리키고자 하는 변수의 자료형에 따라 적절한 타입의 포인터 변수를 선언해야 합니다.

```
int main(void)
{
    int* a;        // a라는 이름의 int형 포인터 변수
    char* b;       // b라는 이름의 char형 포인터 변수
    double* c;     // c라는 이름의 double형 포인터 변수
    ...
}
```

포인터 변수를 선언할 때에는 * 연산자를 사용합니다.

- *: Asterisk, 애스터리스크, 별표
  라틴어로 별을 의미합니다.

참고

✓ 포인터 변수 선언에서 *의 위치에 따른 차이는 없습니다.
✓ 즉, 다음 세 문장은 모두 동일한 포인터 변수의 선언문입니다.

① int* a // 가독성이 높이 때문에 주로 사용합니다.
② int *a
③ int * a

01. 포인터란 무엇인가?

❖ 포인터 변수와 & 연산자 맛보기

① int형 변수 num을 선언하고, 정수 7을 저장시키세요.

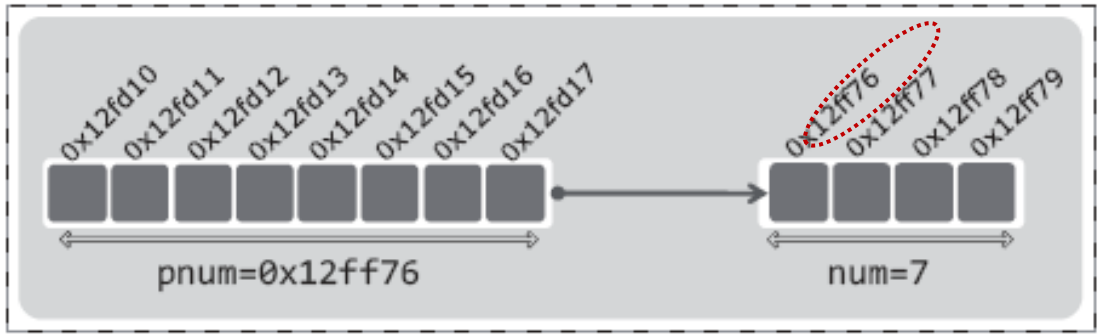
② 이 변수의 주소 값 저장을 위한 포인터 변수 pnum을 선언하세요.

③ 그리고 나서 pnum에 변수 num의 주소 값을 저장하세요.

```
int main(void)
{
      int num = 7;
      int* pnum;        // 포인터 변수 pnum의 선언
      pnum = &num;      // num의 주소 값을 pnum에 저장
      ...
}
```

코드로 옮긴 결과

메모리 저장 상태



"포인터 변수 pnum이 변수 num을 가리킨다"라고 표현합니다.

[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

01. 포인터란 무엇인가?

❖ 포인터 변수의 크기

- 포인터 변수의 크기는 시스템의 주소 값 크기에 따라서 다릅니다.
- 16비트 시스템 → 주소 값 크기 16비트 → 포인터 변수의 크기 16비트!
- 32비트 시스템 → 주소 값 크기 32비트 → 포인터 변수의 크기 32비트!
- 64비트 시스템 → 주소 값 크기 64비트 → 포인터 변수의 크기 64비트!

```
1   #include <stdio.h>
2   int main(void)
3   {
4         int* a;
5         char* b;
6         double* c;
7
8         printf("int형 포인터 변수의 크기: %d바이트\n", sizeof(a));
9         printf("char형 포인터 변수의 크기: %d바이트\n", sizeof(b));
10        printf("double형 포인터 변수의 크기: %d바이트\n", sizeof(c));
11
12        return 0;
13  }
```

```
int형 포인터 변수의 크기: 8바이트
char형 포인터 변수의 크기: 8바이트
double형 포인터 변수의 크기: 8바이트
```

8바이트는 64비트입니다.

02. 포인터와 관련 있는 & 연산자와 * 연산자

01. 포인터란 무엇인가?
03. 연습 문제

02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 변수의 주소 값을 반환하는 & 연산자 (1/2)

- & 연산자는 변수의 주소 값을 반환하므로 상수가 아닌 변수가 피연산자이어야 합니다.
- & 연산자의 반환 값은 포인터 변수에 저장합니다.

```
int main(void)
{
     int num = 5;
     int* pnum = &num;

     ...

     return 0;
}
```

"&num"은 변수 num의 주소 값을 반환하라는 의미를 갖습니다.

02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 변수의 주소 값을 반환하는 & 연산자 (2/2)

- num1은 int형 변수이므로 pnum1은 int형 포인터 변수이어야 합니다.
- num2는 double형 변수이므로 pnum2는 double형 포인터 변수이어야 합니다.

```
int main(void)
{
     int num1 = 5;
     double* pnum1 = &num1; // 형 (Type)이 일치하지 않음!

     double num2 = 5;
     int* pnum2 = &num2;    // 형 (Type)이 일치하지 않음!

     ...

     return 0;
}
```

02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 포인터가 가리키는 메모리를 참조하는 * 연산자 (1/2)

- *pnum은 num을 의미합니다.
- 따라서 num을 놓을 자리에 *pnum을 놓을 수 있습니다.

```
1  #include <stdio.h>
2  int main(void)
3  {
4       int num = 10;
5       int* pnum = &num;       // pnum이 num을 가리킵니다.
6       *pnum = 20;             // pnum이 가리키는 공간(변수)에 20을 저장합니다.
7
8       printf("%d\n", *pnum); // pnum이 가리키는 공간(변수)에 저장된 값을 출력합니다.
9       return 0;
10 }
```

```
20
```

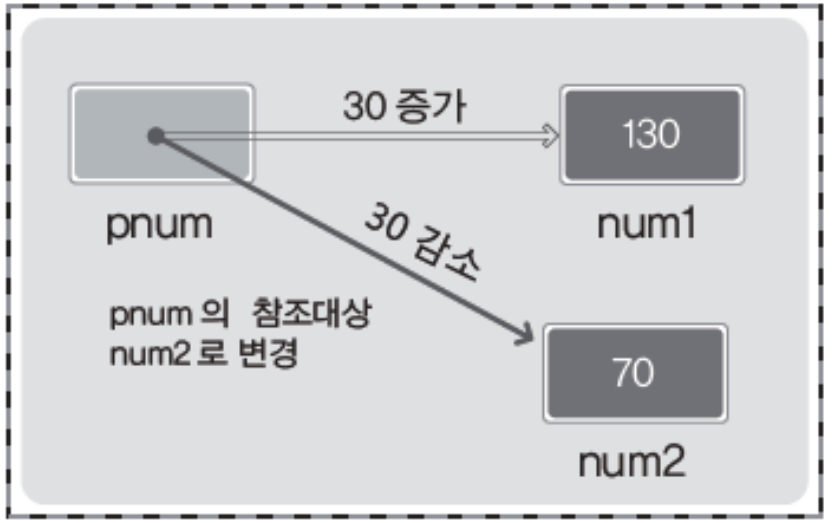
02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 포인터가 가리키는 메모리를 참조하는 * 연산자 (2/2)

```
1  #include <stdio.h>
2  int main(void)
3  {
4       int num1 = 100, num2 = 100;
5       int* pnum;
6
7       pnum = &num1;           // 포인터 pnum이 num1을 가리킵니다.
8       (*pnum) += 30;          // num1 += 30;과 동일합니다.
9
10      pnum = &num2;           // 포인터 pnum이 num2를 가리킵니다.
11      (*pnum) -= 30;          // num2 -= 30;과 동일합니다.
12
13      printf("num1: %d, num2: %d\n", num1, num2);
14      return 0;
15 }
```

**포인터 변수 앞에 * 연산자를 붙이게 되면,
포인터 변수가 가리키는 메모리 공간에 존재하는 값을
참조하라는 의미입니다.**

```
130, 70
```



[사진출처] 윤성우의 열혈 C 프로그래밍 (개정판) (출판사: 오렌지미디어)

02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 다양한 포인터 형(Type)이 존재하는 이유 (1/2)

- 포인터 형은 메모리 공간을 참조하는 방법의 힌트가 됩니다.
- 다양한 포인터 형을 정의한 이유는 * 연산을 통한 메모리의 접근기준을 마련하기 위함입니다.

◆ int형 포인터 변수로 * 연산을 통해 메모리(변수) 접근 시

➢ 4바이트 메모리 공간에 부호 있는 정수의 형태로 데이터를 읽고 씁니다.

◆ double형 포인터 변수로 * 연산을 통해 메모리(변수) 접근 시

➢ 8바이트 메모리 공간에 부호 있는 실수의 형태로 데이터를 읽고 씁니다.

02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 다양한 포인터 형(Type)이 존재하는 이유 (2/2)

```
1  #include <stdio.h>
2  int main(void)
3  {
4       double num = 3.14;
5       int* pnum = &num;       // 형 불일치! 컴파일은 됩니다.
6
7       printf("%d\n", pnum);
8
9       return 0;
10 }
```

```
1226045816
```

결과 값은 매번 달라집니다.

- ✓ pnum이 가리키는 것은 double형 변수인데, pnum이 int형 포인터 변수이므로 int형 데이터처럼 해석!
- ✓ 주소 값이 정수임에도 불구하고 int형 변수에 저장하지 않는 이유는, int형 변수에 저장하면 메모리 공간의 접근을 위한 * 연산이 불가능하기 때문입니다.

02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 잘못된 포인터의 사용과 널 포인터 (1/2)

위험한 코드 ①

```
1  int main(void)
2  {
3       int* ptr;
4       *ptr = 200;
5       return 0;
6  }
```

- ✓ ptr이 쓰레기 값으로 초기화됩니다.
- ✓ 따라서 200이 저장되는 위치가 어디인지 알 수가 없습니다.
- ✓ 매우 위험합니다.
- ✓ 다행히 요즘 운영체제는 이러한 문제를 미리 파악하고서 이와 같은 코드 실행은 허용하지 않고 중지시킵니다.

만약 ptr이 아주 중요한 메모리 공간을 가리키고 있었다면, 시스템 전체에 심각한 문제가 발생할 수도 있습니다.

위험한 코드 ②

```
1  int main(void)
2  {
3       int* ptr = 125;
4       *ptr = 10;
5       return 0;
6  }
```

- ✓ 포인터 변수를 선언과 동시에 초기화를 시켜주는 것이 좋다고 해서 3번째 줄과 같이 초기화를 하면 안됩니다.
- ✓ ptr에 125를 저장했는데, 저장된 위치가 어디인지 알 수 가 없습니다.
- ✓ 매우 위험합니다.

02. 포인터와 관련 있는 & 연산자와 * 연산자

❖ 잘못된 포인터의 사용과 널 포인터 (2/2)

안전한 코드

```
1  int main(void)
2  {
3       int* ptr1 = 0;
4       int* ptr2 = NULL;
5       ...
6  }
```

- ✓ 잘못된 포인터 연산을 막기 위해서 특정한 값으로 초기화하지 않는 경우에는 널 포인터로 초기화하는 것이 안전합니다.
- ✓ 널 포인터 NULL은 숫자 0을 의미합니다.
- ✓ 그리고 0은 0번지를 뜻하는 것이 아니라, 아무것도 가리키지 않는다는 의미로 해석이 됩니다.

03. 연습 문제

01. 포인터란 무엇인가?
02. 포인터와 관련 있는 & 연산자와 * 연산자

03. 연습 문제

❖ 연습 문제 1.

- 다음 프로그램 실행 시 포인터와 변수와의 관계를 나름대로 그림으로 그려서 설명해 보세요. 또한 출력 결과도 예상해 보세요.

```
1   /* example1.c */
2   #include <stdio.h>
3
4   int main(void)
5   {
6         int n = 10;
7         int* p1 = &n;
8         int* p2 = p1;
9
10        printf("%d ", (*p1)++);
11        printf("%d ", (*p2)++);
12        printf("%d\n", n);
13
14        return 0;
15  }
```

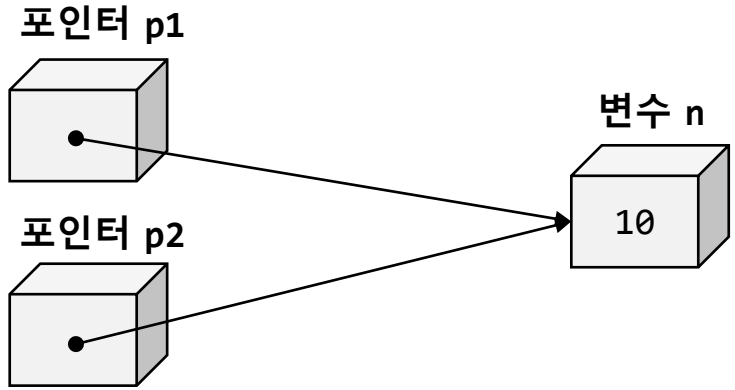
03. 연습 문제

❖ 연습 문제 1. 정답 및 해설

```
1   /* example1.c */
2   #include <stdio.h>
3
4   int main(void)
5   {
6         int n = 10;
7         int* p1 = &n;
8         int* p2 = p1;
9
10        printf("%d ", (*p1)++);
11        printf("%d ", (*p2)++);
12        printf("%d\n", n);
13
14        return 0;
15  }
```

```
10 11 12
```

✓ 8번째 줄까지 실행하게 되면 다음과 같은 메모리 구조가 형성됩니다.

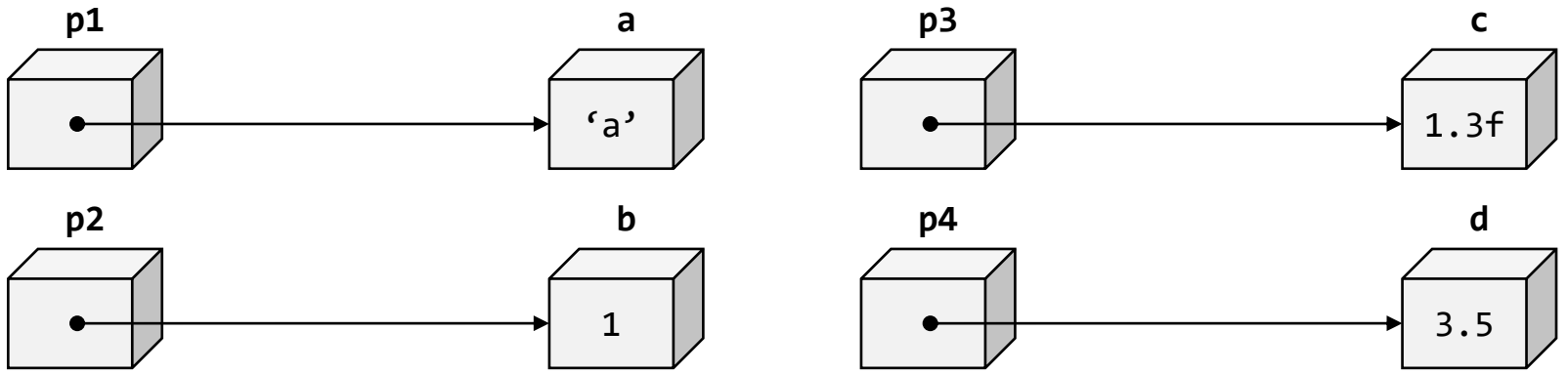


✓ 10번째 줄에서 p1이 가리키는 변수의 값을 출력 후 1 증가시킵니다.
✓ 따라서 10이 출력될 것이고, 변수 n은 11이 됩니다.
✓ 11번째 줄에서도 p2가 가리키는 변수의 값을 출력 후 1 증가시키므로 11이 출력되고 나서, 변수 n은 12가 될 것입니다.
✓ 마지막으로 12번째 줄에서는 n을 출력하므로 12가 출력됩니다.

03. 연습 문제

❖ 연습 문제 2.

- char형 변수 a, int형 변수 b, float형 변수 c, double형 변수 d, 이렇게 총 4개의 변수를 선언하세요. (초기값은 임의로 설정하세요.) 그리고 포인터 변수 p1, p2, p3, p4를 선언해서 각각 a, b, c, d를 가리키게 하세요. 다음 그림은 지금 이야기한 상황을 보충 설명해 줍니다.



- 그 다음에 포인터 p1, p2, p3, p4를 이용해서 a, b, c, d의 값을 1 증가시키세요. 그리고 나서 a, b, c, d의 값을 출력해 보세요.

03. 연습 문제

❖ 연습 문제 2. 정답 및 해설

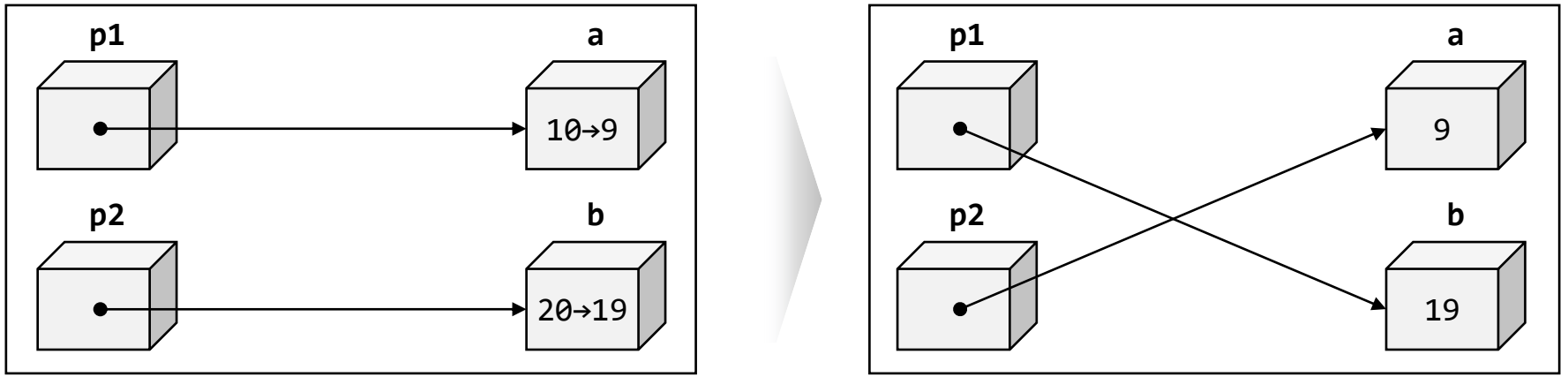
```
1   /* example2.c */
2   #include <stdio.h>
3
4   int main(void)
5   {
6         char a = ‘a’;
7         int b = 1;
8         float c = 1.3f;
9         double d = 3.5;
10
11        char* p1 = &a;
12        int* p2 = &b;
13        float* p3 = &c;
14        double* p4 = &d;
15
16        *p1 = *p1 + 1;   // *p1 += 1;과 같습니다.
17        *p2 = *p2 + 1;   // *p2 += 1;과 같습니다.
18        *p3 = *p3 + 1;   // *p3 += 1;과 같습니다.
19        *p4 = *p4 + 1;   // *p4 += 1;과 같습니다.
20
21        printf(“a: %c\n”, *p1);
22        printf(“b: %d\n”, *p2);
23        printf(“c: %f\n”, *p3);
24        printf(“d: %f\n”, *p4);
25
26        return 0;
27  }
```

```
a: b
b: 2
c: 2.300000
d: 4.500000
```

03. 연습 문제

❖ 연습 문제 3.

- int형 변수 a와 b를 선언과 동시에 각각 10, 20으로 초기화시키세요. 그리고 포인터 변수 p1, p2를 선언한 다음 각각 변수 a와 b를 가리키게 하세요. 이러한 상태에서 간접 접근 방식에 의해서 값을 1씩 감소시키세요. 그 다음 포인터 변수 p1과 p2가 가리키는 대상을 서로 바꿔주세요.
  지금까지의 상황을 그림으로 설명하면 다음과 같습니다.



- 최종적으로 p1과 p2가 가리키는 변수의 값을 출력해서 포인터가 가리키는 대상이 변경되었는지 확인할 수 있도록 프로그램을 작성하세요.

03. 연습 문제

❖ 연습 문제 3. 정답 및 해설

```c
/* example3.c */
#include <stdio.h>

int main(void)
{
	int a = 10;
	int b = 20;

	int* p1 = &a;
	int* p2 = &b;
	int* temp = NULL;

	(*p1)--;
	(*p2)--;

	// p1과 p2가 가리키는 대상을 서로 변경
	temp = p1;
	p1 = p2;
	p2 = temp;

	printf("%d, %d\n", *p1, *p2);
	return 0;
}
```

```
19
9
```

끝맺음

- ❖ 01. 포인터란 무엇인가?
- ❖ 02. 포인터와 관련 있는 & 연산자와 * 연산자
- ❖ 03. 연습 문제

THANK YOU!
Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr