9강

포인터

동양미래대학교 강환수 교수

본 강의 사용 및 참조 자료

- Perfect C, 3판, 강환수 외 2인 공저, 인피니티북스, 2021

11장 포인터 기초

목차

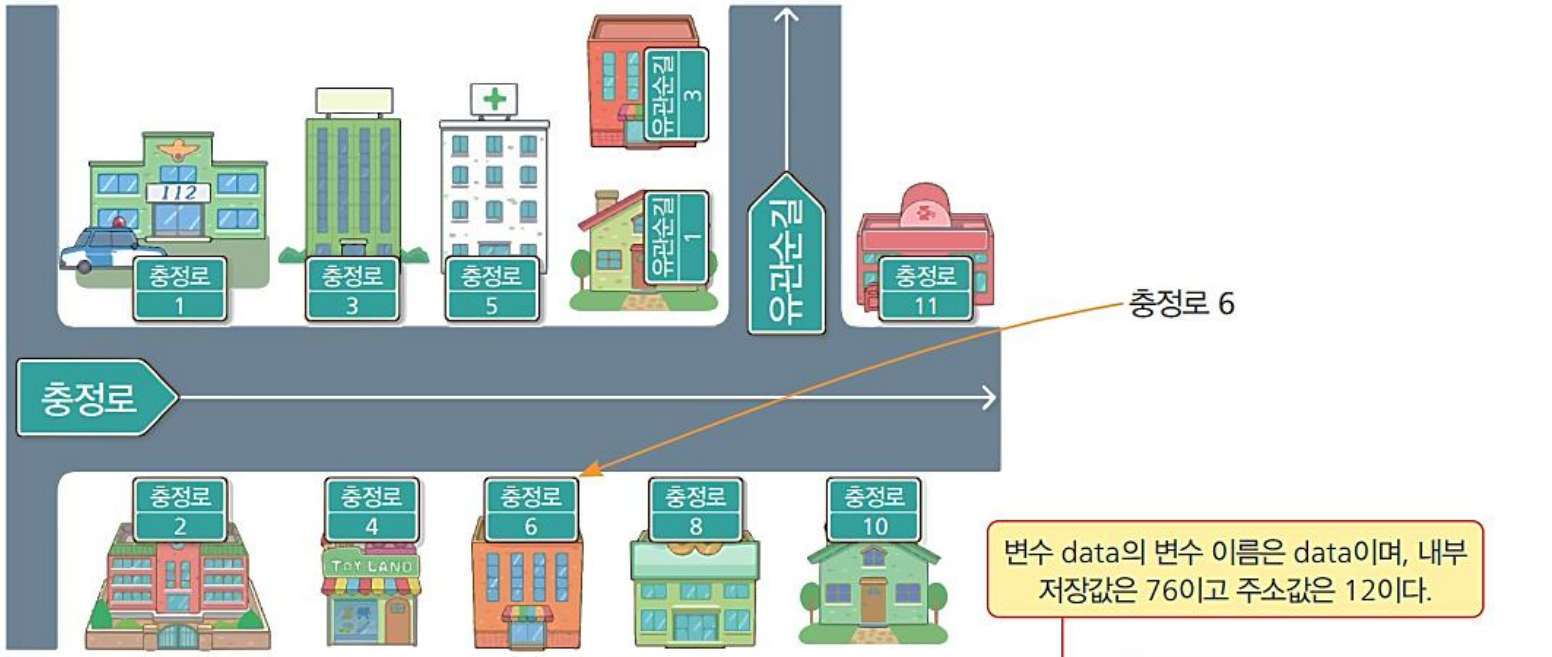
01

포인터 변수와 선언

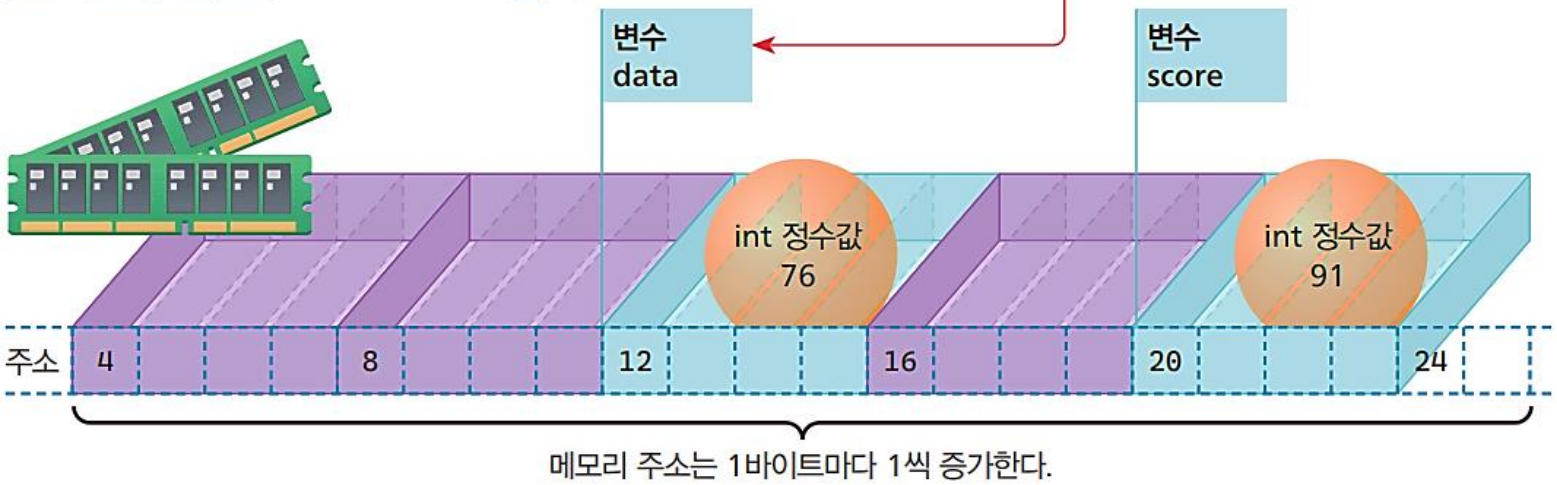
주소 개념

- 고유한 주소(address)
  - 메모리 공간은 8비트인 1 바이트마다 고유한 숫자
    - 0부터 바이트마다 1씩 증가
  - 메모리 주소는 저장 장소인 변수이름과 함께 기억장소를 참조하는 또 다른 방법

일상 주소와 메모리 주소



변수 data의 변수 이름은 data이며, 내부 저장값은 76이고 주소값은 120이다.



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 462쪽

주소 연산자 &

- &(ampersand)
  - 피연산자인 변수의 메모리 주소를 반환
  - 함수 scanf()에서
    - 일반 변수 앞에는 주소연산자 &를 사용

변수의 주소 값 출력

- 윈도 10의 64비트 시스템 주소 값
  - 8바이트(64비트)
    - 16진수(4bit)의 16개 자릿수로 출력
      - 4×16 == 64
- 형식제어문자 %p로 직접 출력
- 형식제어문자 %llu로 출력
  - 64비트의 0과 양수의 정수형을 위한 형식 제어문자
  - %llu는 long long unsigned를 의미
    - 64비트 정수 출력: %ll 사용

변수의 주소 값

- 자료형 uintptr_t
  - 헤더파일 vadefs.h에 다음으로 정의
    - typedef unsigned __int64 uintptr_t;
    - uintptr_t를 unsigned __int64와 동일한 자료형으로 정의
  - __int64는 long long int와 같이 64비트 정수 자료형

연산자 sizeof(&input)의 반환 값

- 자료형 size_t 유형의 주소 크기, 형식제어문자 %zu로 출력
  - z는 size, u는 unsigned를 의미
- 자료형 size_t: unsigned long long
- 연산자 sizeof의 반환 값
  - %zd로도 가능

포인터 변수 선언

- 포인터 변수, 간단히 포인터
- 변수 선언
  - 자료형과 포인터 변수 이름 사이에 연산자 *(asterisk)를 삽입
  - int *ptrint
    - ‘int 포인터 ptrint’라고 읽음
  - 포인터 변수 선언

    자료형 *변수이름 ;

```
int *ptrint;
short*ptrshort;
char * ptrchar;
double *ptrdouble;
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 466쪽

포인터에 주소 값 저장

- 어느 변수의 주소 값을 저장하려면
  - 반드시 그 변수의 자료유형과 동일한 포인터 변수에 저장

```
int data = 100;
int *ptrint;

ptrint = &data;
```

변수 data를 선언해 100을 저장

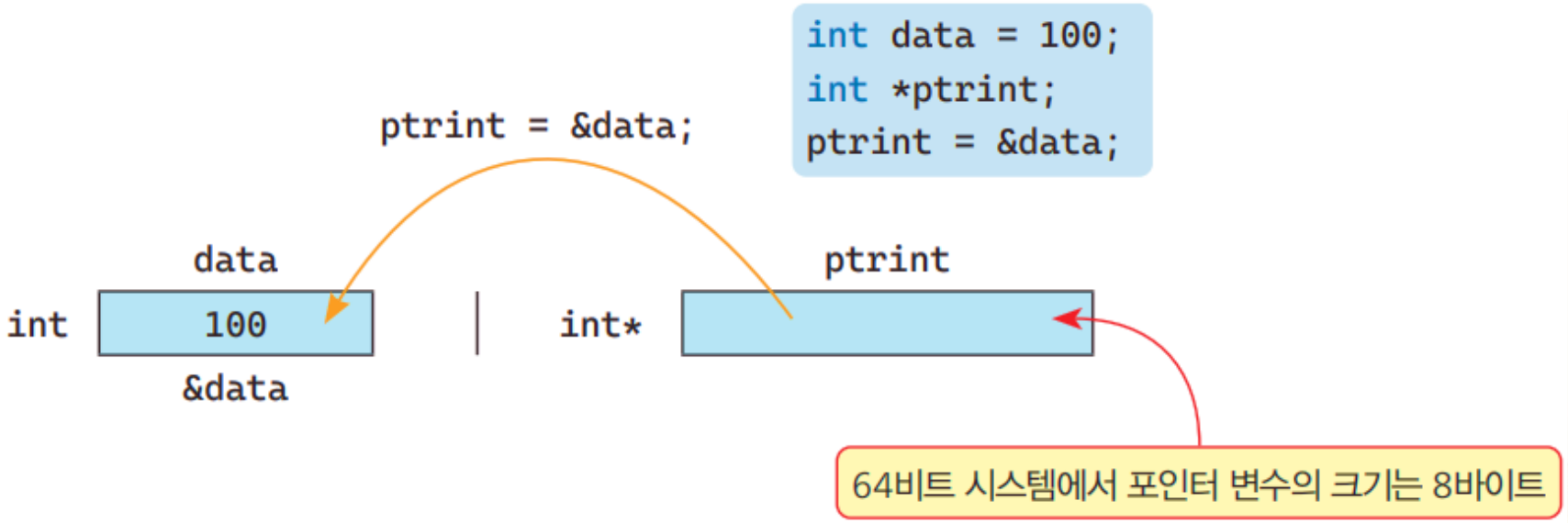
변수 ptrint는 정수 int형 변수의 주소를 저장하는 포인터 변수이며, 이 문장으로 포인터 변수 ptrint를 선언만 해 쓰레기 주소값이 저장

변수 ptrint에 data의 주소 값을 저장하는 문장

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 466쪽

포인터에 주소 값 저장

- '포인터 변수 ptrint는 변수 data를 가리킨다'
  - 또는 '참조(reference)한다'라고 표현
- 64비트 시스템인 윈도 10에서 포인터 변수
  - 가리키는 변수의 종류에 관계없이 크기가 모두 8바이트

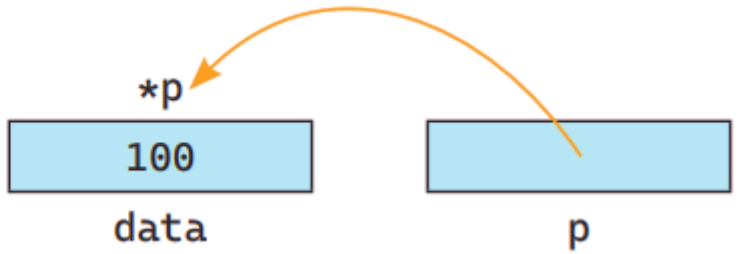


[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 467쪽

단항 연산자인 간접연산자 *

- 포인터 변수가 갖는 주소로 그 주소의 원래 변수를 참조하는 연산자와 방법
  - 간접연산자(indirection operator) *를 사용한 역참조
    - 전위연산자(피연산자 앞에 위치)로 피연산자는 포인터
    - *p는 피연산자인 p가 가리키는 변수 자체를 반환
  - 포인터 p는 data의 주소 값을 가지므로 *p는 data와 같음

```
int data = 100;
int *p = &data;
printf("간접참조 출력: %d \n", *p);
```

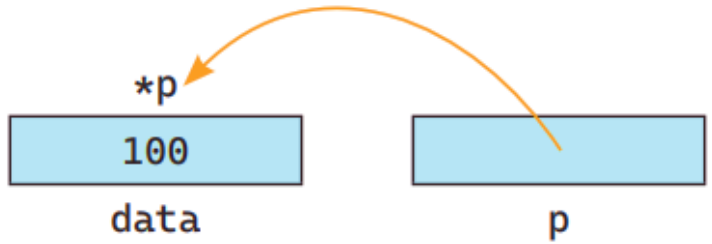


[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 472쪽

포인터 p가 가리키는 변수가 data

- *p은 변수 data를 의미
  - 변수 data로 가능한 작업은 *p로도 가능
  - 문장 *p = 200;으로 변수 data의 저장 값을 200으로 수정 가능

```
int data = 100;
int *p = &data;
printf("간접참조 출력: %d \n", *p);
```



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 472쪽

실습예제

Prj04 04dereference.c 포인터 변수와 간접연산자 *를 이용한 간접참조 난이도: ★

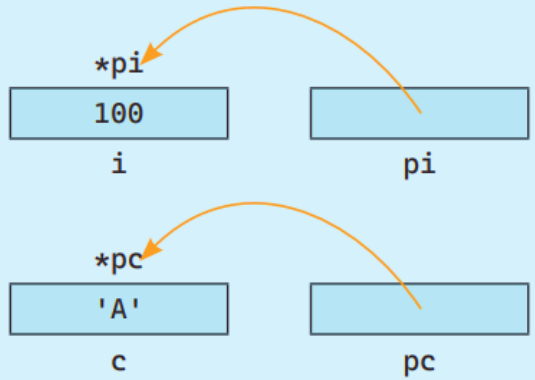
```c
#include <stdio.h>

int main(void)
{
    int i = 100;
    char c = 'A';

    int *pi = &i;
    char *pc = &c;
    printf("간접참조 출력: %d %c\n", *pi, *pc);

    *pi = 200; //변수 i를 *pi로 간접참조하여 그 내용을 수정
    *pc = 'B'; //변수 c를 *pc로 간접참조하여 그 내용을 수정
    printf("직접참조 출력: %d %c\n", i, c);

    return 0;
}
```




```
간접참조 출력: 100 A
직접참조 출력: 200 B
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 473쪽

02

포인터 형변환과 다중 포인터

변수의 내부 저장 표현

- 변수 value에 16진수 0x61626364를 저장
  - 변수 value의 주소가 100번지
    - 100번지 1바이트 내부에 16진수 64가 저장
      - 다음 주소 101번지에는 63이 저장
      - 다음에 각각 62, 61이 저장
    - 즉 자연스럽게 0x61626364의 수가 큰 주소 값에서 작은 주소 값으로 저장
    - 반환 주소 값은 가장 작은 주소 값에 해당

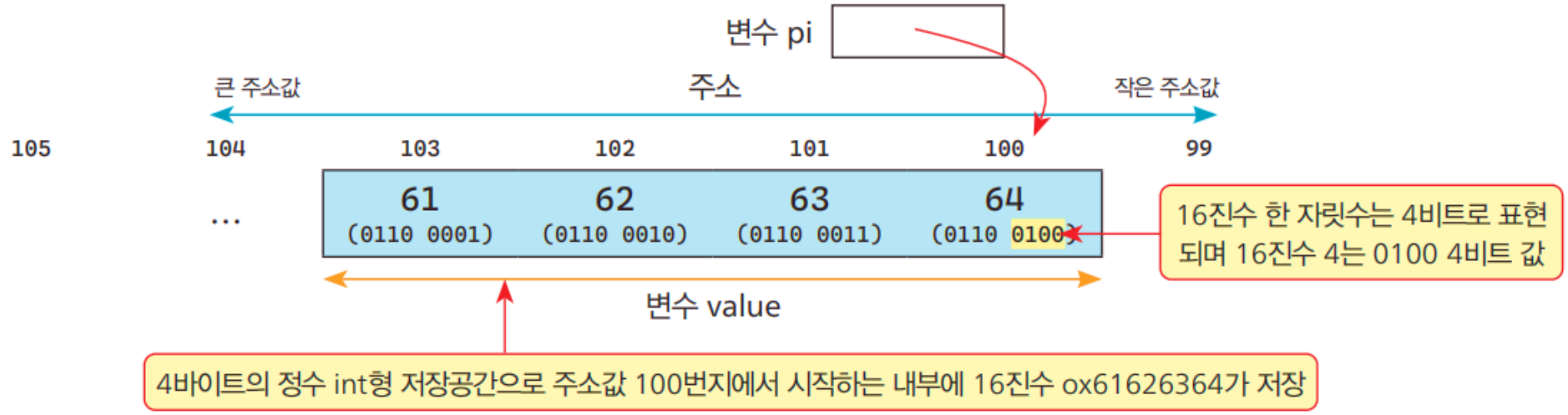
int 형: 32비트(== 8×4 비트)

16진수 한 자릿수: 4비트

```
int value = 0x61626364; // 정수의 일부분인 코드 61은 문자 'a'
int *pi = &value;

printf("%#x %d\n", value, value);
```

다음이 출력으로 10진수 값은 16진수 0x61626364에 해당하는 10진수 0x61626364 1633837924



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 478쪽

명시적 형변환

- 포인터 변수는 동일한 자료형끼리만 대입이 가능
  - 만일 대입문에서 포인터의 자료형이 다르면 경고가 발생
- 포인터 변수는 자동으로 형변환(type cast)이 불가능
  - 필요하면 명시적으로 형변환을 수행

```
int value = 0x44434241; // 정수의 일부분인 코드 41은 문자 'A'
int *pi = &value;
char *pc = &value;
```

warning C4133: '초기화중' : 'int *'과(와) 'char *' 사이의 형식이 호환되지 않습니다.

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 479쪽

포인터 변수와 간접 참조

포인터 변수

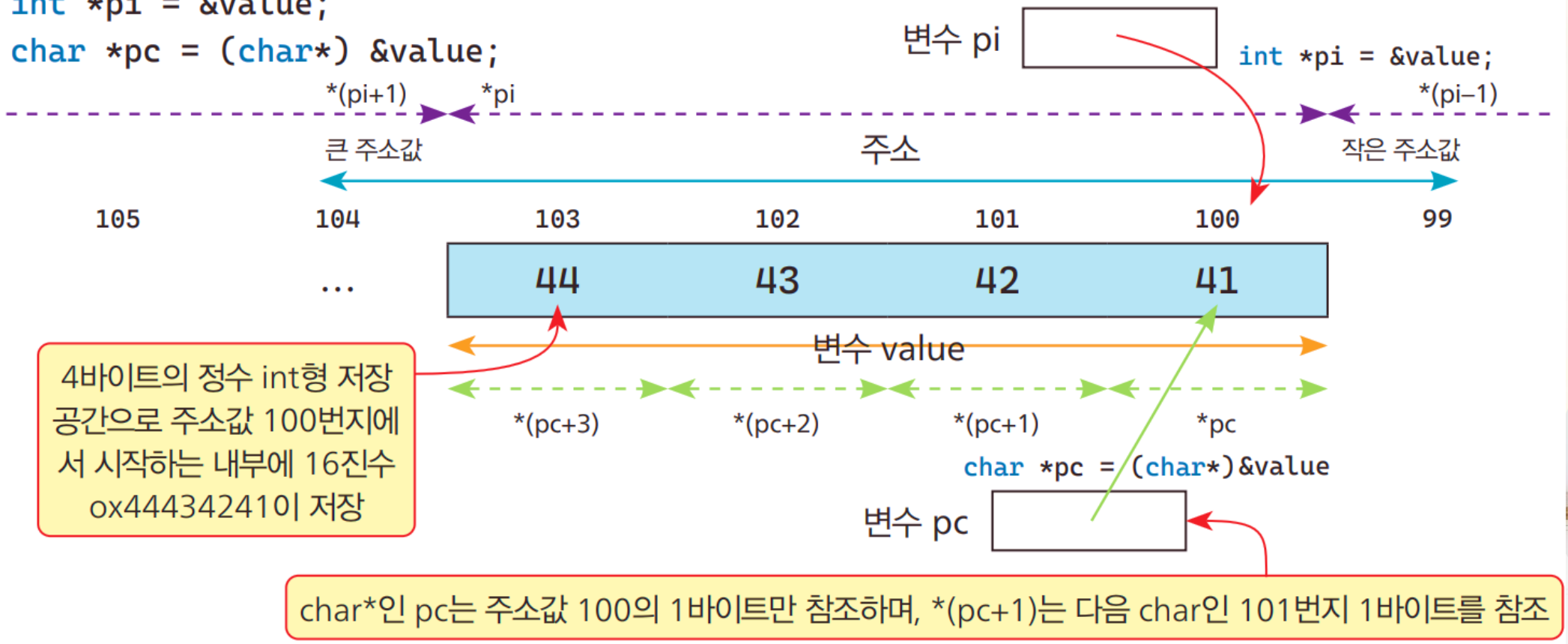
- 지정된 주소 값을 시작하여 그 변수 자료형의 크기만큼 저장공간을 참조
- 동일한 메모리의 내용과 주소로부터 참조하는 값이 포인터의 자료형에 따라 달라짐

*pc로 수행하는 간접 참조

- pc가 가리키는 주소에서부터 1바이트 크기의 char 형 자료를 참조
- *pi 는 4바이트인 정수 0x44434241
- *pc는 1바이트인 문자코드 0x41을 참조

포인터 변수와 명시적 형변환

```
int value = 0x44434241; // 정수의 일부분인 코드 41은 문자 'A'
int *pi = &value;
char *pc = (char*) &value;
```



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 479쪽

실습예제 1/2

Prj06 06ptypecast.c 포인터 자료형의 변환 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      //int value = 0x61626364;     // 정수의 일부분인 코드 61은 문자 'a'
06      int value = 0x44434241;       // 정수의 일부분인 코드 41은 문자 'A'
07      printf("저장 값: %#x(16진수) %d(10진수)\n\n", value, value);
08
09      int *pi = &value;
10      char *pc = (char*) &value;
11
12      printf("변수명    저장값        주소값\n");
13      printf("----------------------\n");
14      printf(" value  %#x  %llu\n\n", value, (uintptr_t)pi); // 정수 int형 출력
15
16      printf("간접참조 코드 문자    주소값\n");
17      printf("----------------------\n");
```

4바이트인 10진수를 1바이트씩만 분리해서 출력한다면 16진수 44는 문자 'D'에 해당

char 포인터 pc를 선언하여 int 변수 value의 주소를 char 포인터로 형변환하여 저장, 이제 pc는 char 포인터이므로 1바이트씩 이동 가능

주소값을 10진수로 출력하려면 (uintptr_t)로 변환해 출력

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 480쪽

실습예제 2/2

```
19      for (int i = 0; i <= 3; i++)
20      {
21          char ch = *(pc + i);
22          printf(" *(pc+%d) %#x %3c %llu\n", i, ch, ch, (uintptr_t)(pc + i));
23      }
24
25      return 0;
26  }
```

char 변수 ch에 (pc+i)가 가리키는 문자를 저장하며, i가 0에서 3까지 반복되므로 pc가 가리키는 문자에서부터 이웃한 3개, 총 4개 문자를 순서로 대입

```
저장 값: 0x44434241(16진수) 1145258561(10진수)

  변수명        저장값        주소값
--------------------------------
  value    0x44434241   287588219092

  간접참조    코드  문자      주소값
--------------------------------
 *(pc+0)   0x41    A    287588219092
 *(pc+1)   0x42    B    287588219093
 *(pc+2)   0x43    C    287588219094
 *(pc+3)   0x44    D    287588219095
```

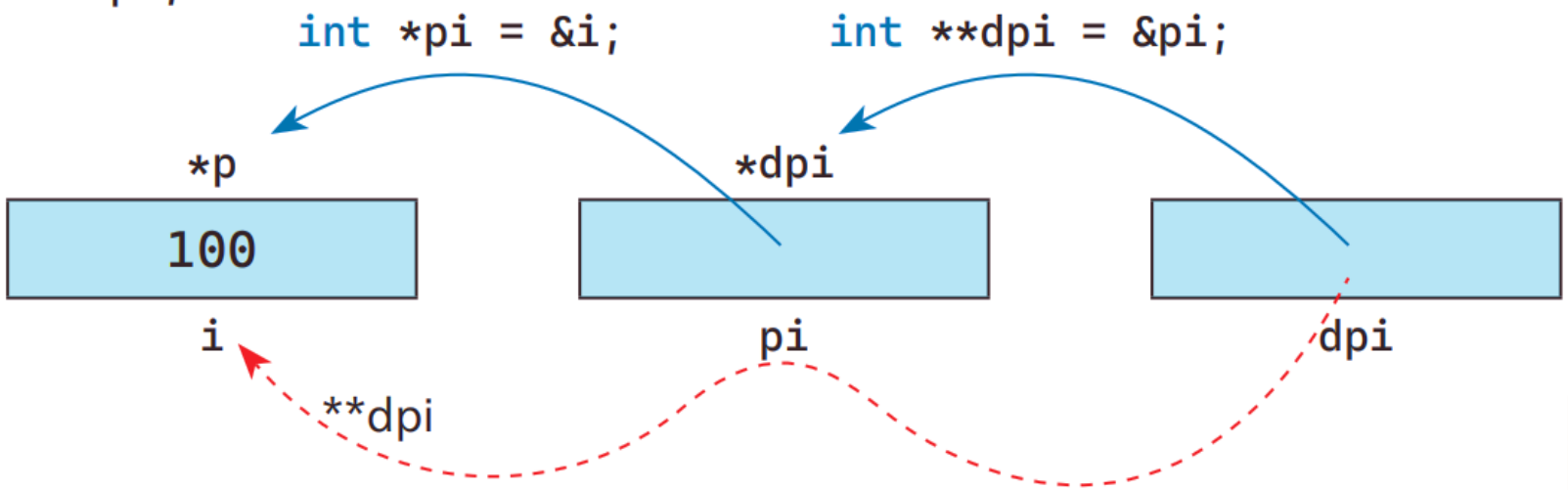
[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 480쪽

이중 포인터

- 포인터 변수의 주소 값을 갖는 변수
- 삼중 포인터
  - 이중 포인터의 주소 값을 갖는 변수
- 모두 다중 포인터
  - 변수 선언에서 *를 여러 번 이용하여 다중 포인터 변수를 선언
    - pi는 포인터
    - 포인터 변수 pi의 주소 값을 저장하는 변수 dpi는 이중 포인터

이중 포인터의 메모리와 변수

```
int i = 20;
int *pi = &i;
int **dpi = &pi;
```




```
*pi = i + 2;        // i = i + 2;
**dpi = *pi + 2;    // i = i + 2;
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 481쪽

03

포인터를 사용한 배열 활용

배열이름을 이용한 참조 1/2

- 배열 score
  - 배열이름 score 자체가 배열 첫 원소의 주소값인 상수
  - 연산식 (score + 1)
    - 배열의 두 번째 원소의 주소값
  - 일반화하면 (score + i)
    - 배열의 (i + 1)번째 원소 주소
- 간접연산자로 사용한 *(score + i)
  - 배열의 (i+1) 번째 원소값으로 score[i]와 동일

배열이름을 이용한 참조 2/2

```
int score[] = {10, 20, 30};
```

주소값 참조	&score[0] score	&score[1] score+1	&score[2] score+2
배열 score	5	10	15
저장 값 참조	score[0] *score	score[1] *(score+1)	score[2] *(score+2)

- 배열의 주소값(배열 첫 번째 원소의 주소값): score, &score[0]
- 배열 첫 번째 원소 저장 값: *score, score[0]
- 배열 (i+1)번째 원소 주소값: (score + i), &score[i]
- 배열 (i+1)번째 원소 저장 값: *(score + i), score[i]

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 488쪽

원소의 주소와 다양한 접근 방법

배열 초기화 문장		int score[] = {10, 20, 30};		
원소 값		10	20	30
배열원소 접근 방법	score[i]	score[0]	score[1]	score[2]
	*(score+i)	*score	*(score+1)	*(score+2)
주소값(첫 주소 + 배열원소 크기*i)		100	104 (100 + 1*4)	108 (100 + 2*4)
주소값 접근 방법	&score[i]	&score[0]	&score[1]	&score[2]
	score+i	score	score+1	score+2

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 489쪽

포인터 배열 개요와 선언

- 포인터 배열(array of pointer)
  - 주소값을 저장하는 포인터를 배열 원소로 하는 배열
  - 일반 배열 선언에서 변수이름 앞에 *를 붙이면 포인터 배열 변수 선언
- int *pa[3]
  - 배열크기가 3인 포인터 배열
  - pa [0]
    - 변수 a의 주소를 저장
  - pa [1] : 변수 b의 주소를 저장
  - pa [2] : 변수 c의 주소를 저장

포인터 배열 개요와 선언

- double *dary[5] = {NULL};
  - NULL 주소를 하나 지정, 나머지 모든 배열원소에 NULL 주소가 지정
  - 문장 float *ptr[3] = {&a, &b, &c};
    - 변수 주소를 하나, 하나 직접 지정하여 저장 가능
  - 포인터 배열 변수 선언

    자료형 *변수이름[배열크기] ;

```
int *pary[5];
char *ptr[4];
float a, b, c;
double *dary[5] = {NULL};
float *ptr[3] = {&a, &b, &c};
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 497쪽

실습예제 1/2

Prj13 13aryptr .c 2차원 배열을 가리키는 배열 포인터의 선언과 이용 난이도: ★★

```c
#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
#include <stdio.h>

#define SIZE 3

int main(void)
{
   //포인터 배열 변수 선언
   int* pary[SIZE] = { NULL };
   int a = 10, b = 20, c = 30;

   pary[0] = &a;
   pary[1] = &b;
   pary[2] = &c;
   for (int i = 0; i < SIZE; i++)
      printf("*pary[%d] = %d\n", i, *pary[i]);

   for (int i = 0; i < SIZE; i++)
   {
      scanf("%d", pary[i]);
      printf("%d, %d, %d\n", a, b, c);
   }
```

정수를 입력하고 엔터 키를 누르면 진행

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 497쪽

실습예제 2/2

```
23
24      return 0;
25  }
```

```
*pary[0] = 10
*pary[1] = 20
*pary[2] = 30
1
1, 20, 30
2
1, 2, 30
3
1, 2, 3
```

정수를 입력하고 엔터 키를 누르면 진행

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 498쪽

정 리 하 기

정리하기

- 메모리는 유일한 주소 값이 있으며
  &변수는 변수의 주소 값을 반환한다.
- 포인터는 주소 값을 저장하는 변수이다.
- 간접연산자(indirection operator) *를 사용한
  *p는 피연산자인 p가 가리키는 변수 자체를 반환한다.
- 포인터 변수는 동일한 자료형끼리만 대입이 가능하다.
- 이중 포인터는 포인터 변수의 주소 값을 갖는 변수이다

정리하기

- 배열 score에서 간접연산자로 사용한
  *(score + i)는 score[i]와 같다.
- 포인터 배열은
  주소값을 저장하는 포인터를 배열 원소로 하는 배열이다.

10강

다음시간안내

문자와 문자열