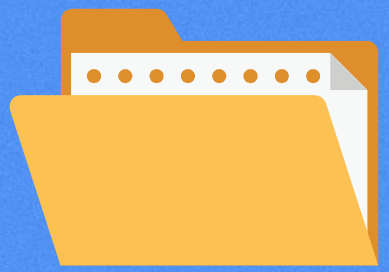
제08장
포인터 기초



01. 포인터 변수와 선언

주소 개념

- **주소(address)**
  - 메모리 공간은 1 바이트마다 순차적인 고유한 번호
  - 메모리 주소는 저장 장소인 변수이름과 함께 기억 장소를 참조하는 또 다른 방법
    - '상허도서관'과 같이 장소이름이 변수명
    - (동경 135.34, 북위37.8) == 메모리 주소
  - 메모리 주소가 왜 필요하지요?
    - 보다 편리하고 융통성 있는 프로그램이 가능
      - 주소값을 이용하여 주소가 가리키는 변수의 값을 참조 가능

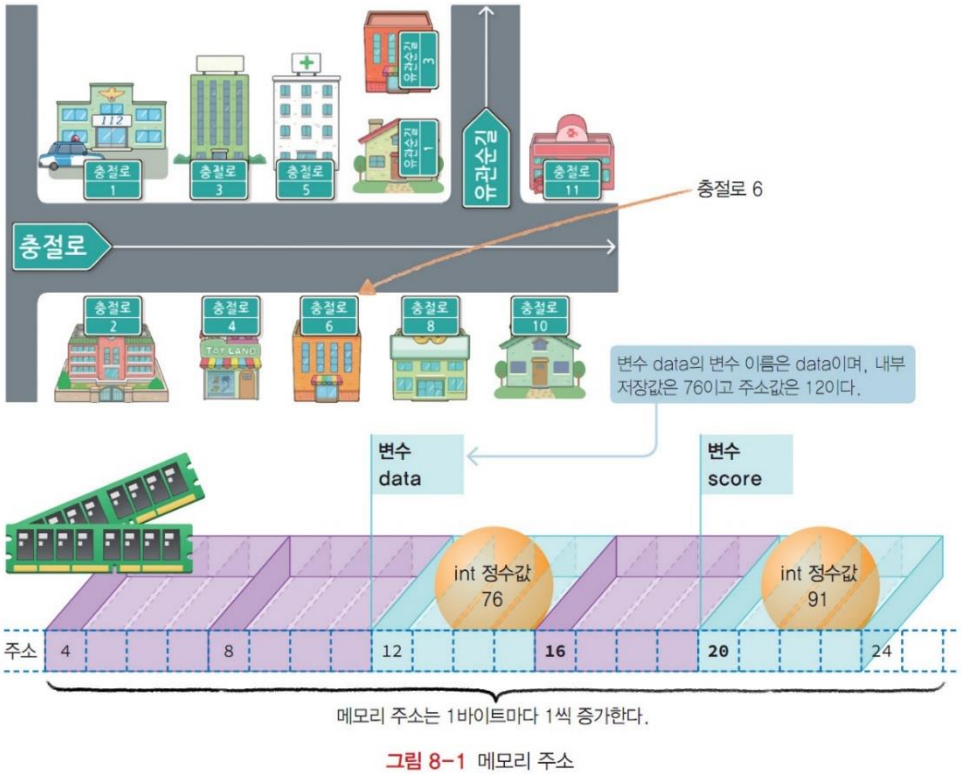


그림 8-1 메모리 주소

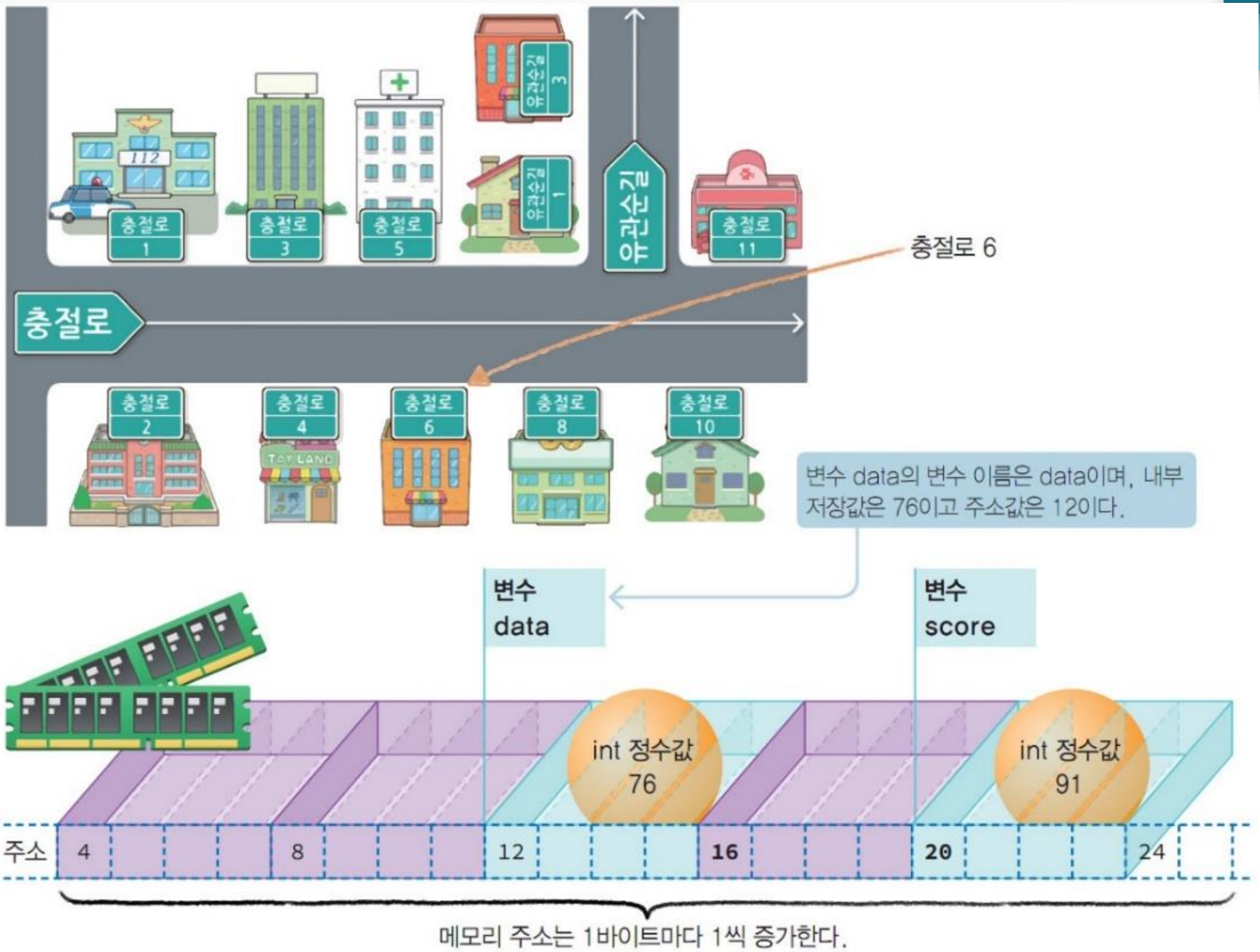

충절로
충절로 6
유관순길
112
충절로 1
충절로 3
충절로 5
유관순길 3
유관순길 1
충절로 11
충절로 2
충절로 4
충절로 6
충절로 8
충절로 10
TOY LAND
변수 data의 변수 이름은 data이며, 내부 저장값은 76이고 주소값은 120이다.
변수 data
변수 score
int 정수값 76
int 정수값 91
주소
4
8
12
16
20
24
메모리 주소는 1바이트마다 1씩 증가한다.

그림 8-1 메모리 주소

주소연산자 &

- **& : 피연산자인 변수의 메모리 주소를 반환하는 주소연산자**
  - 함수 scanf()에서 입력값을 저장하는 변수의 주소값이 인자로써 함수에 넘겨짐
  - 함수 scanf()에서 일반 변수 앞에는 주소연산자 &를 사용

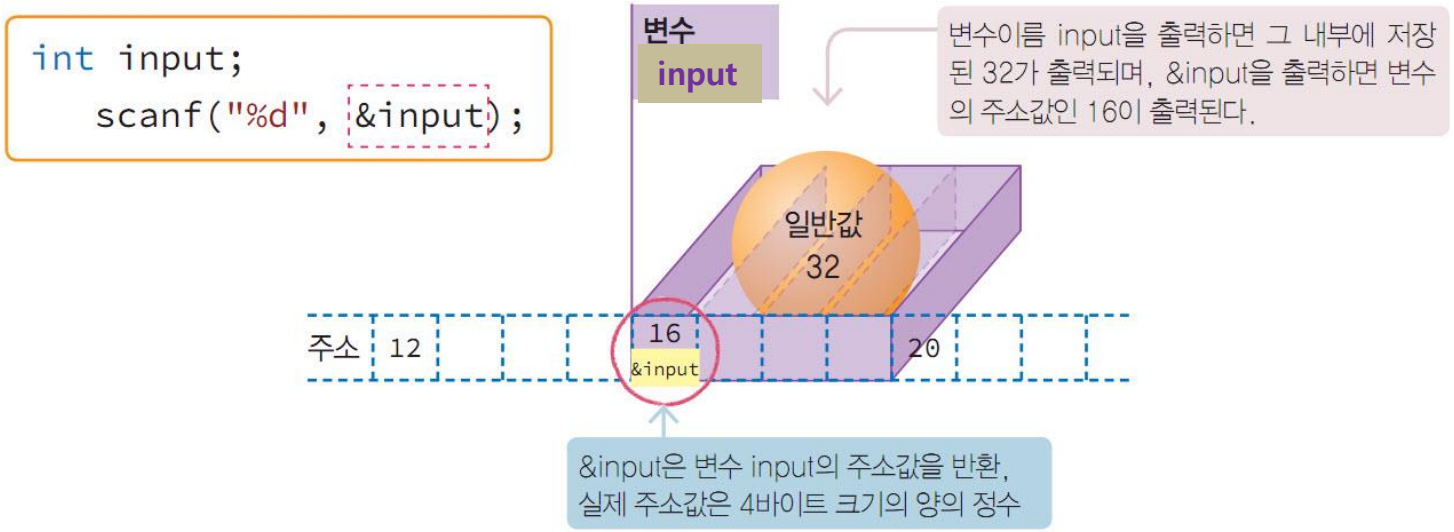


그림 8-2 주소연산자 & 이해

주소값 출력

예제 address.c

- 변수의 값과 주소 값을 출력

- 변수의 주소값 출력
  - 형식제어문자 %u 또는 %d로 직접 출력
  - 최근 비주얼 스튜디오에서는 경고가 발생하니 주소값을 int 또는 unsigned로 변환하여 출력
- 만일 16진수로 출력
  - 형식제어문자 %p를 사용

실습예제 8-1 address.c

```
01  // file: address.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04
05  int main(void)
06  {
07     int input;
08
09     printf("정수 입력: ");
10     scanf("%d", &input);
11     printf("입력값: %d\n", input);
12     printf("주소값: %u(10진수), %p(16진수)\n", (int) &input, &input);
13     printf("주소값: %d(10진수), %#X(16진수)\n", (unsigned) &input,
          (int) &input);
14     printf("주소값 크기: %d\n", sizeof(&input));
15
16     return 0;
17  }
```

설명

```
07  int 형 변수 input 선언
09  저장값 입력을 위한 프롬프트
10  표준입력에서 반드시 &input
12  변수 data의 주소값을 참조하려면 &input을 사용하며, 표준출력에서 %u, %p등 사용
13  변수 data의 주소값을 참조하려면 &input을 사용하며, 표준출력에서 %d, %#X등 사용,
    변환명세에서 필요한 자료값으로 변환
14  주소값의 크기도 4바이트
```

실행결과

```
정수 입력: 100
입력값: 100
주소값: 3799464(10진수), 0039F9A8(16진수)
주소값: 3799464(10진수), 0X39F9A8(16진수)
주소값 크기: 4
```

%p

%#X

실습예제 8-1 address.c

```
01  // file: address.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04
05  int main(void)
06  {
07     int input;
08
09     printf("정수 입력: ");
10     scanf("%d", &input);
11     printf("입력값: %d\n", input);
12     printf("주소값: %u(10진수), %p(16진수)\n", (int) &input, &input);
13     printf("주소값: %d(10진수), %#X(16진수)\n", (unsigned) &input,
          (int) &input);
14     printf("주소값 크기: %d\n", sizeof(&input));
15
16     return 0;
17  }
```

설명

```
07  int 형 변수 input 선언
09  저장값 입력을 위한 프롬프트
10  표준입력에서 반드시 &input
12  변수 data의 주소값을 참조하려면 &input을 사용하며, 표준출력에서 %u, %p등 사용
13  변수 data의 주소값을 참조하려면 &input을 사용하며, 표준출력에서 %d, %#X등 사용,
    변환명세에서 필요한 자료값으로 변환
14  주소값의 크기도 4바이트
```

실행결과

```
정수 입력: 100
```

메모리 주소 저장 변수인 포인터 변수

- **포인터 변수**
  - 주소값을 저장하는 변수
    - 변수의 주소값은 반드시 포인터 변수에 저장
    - (일반 변수에는 일반 자료 값이 저장)
  - 일반 변수와 구별되며 선언방법이 다름

그림 8-3 메모리 주소를 저장하는 포인터 변수

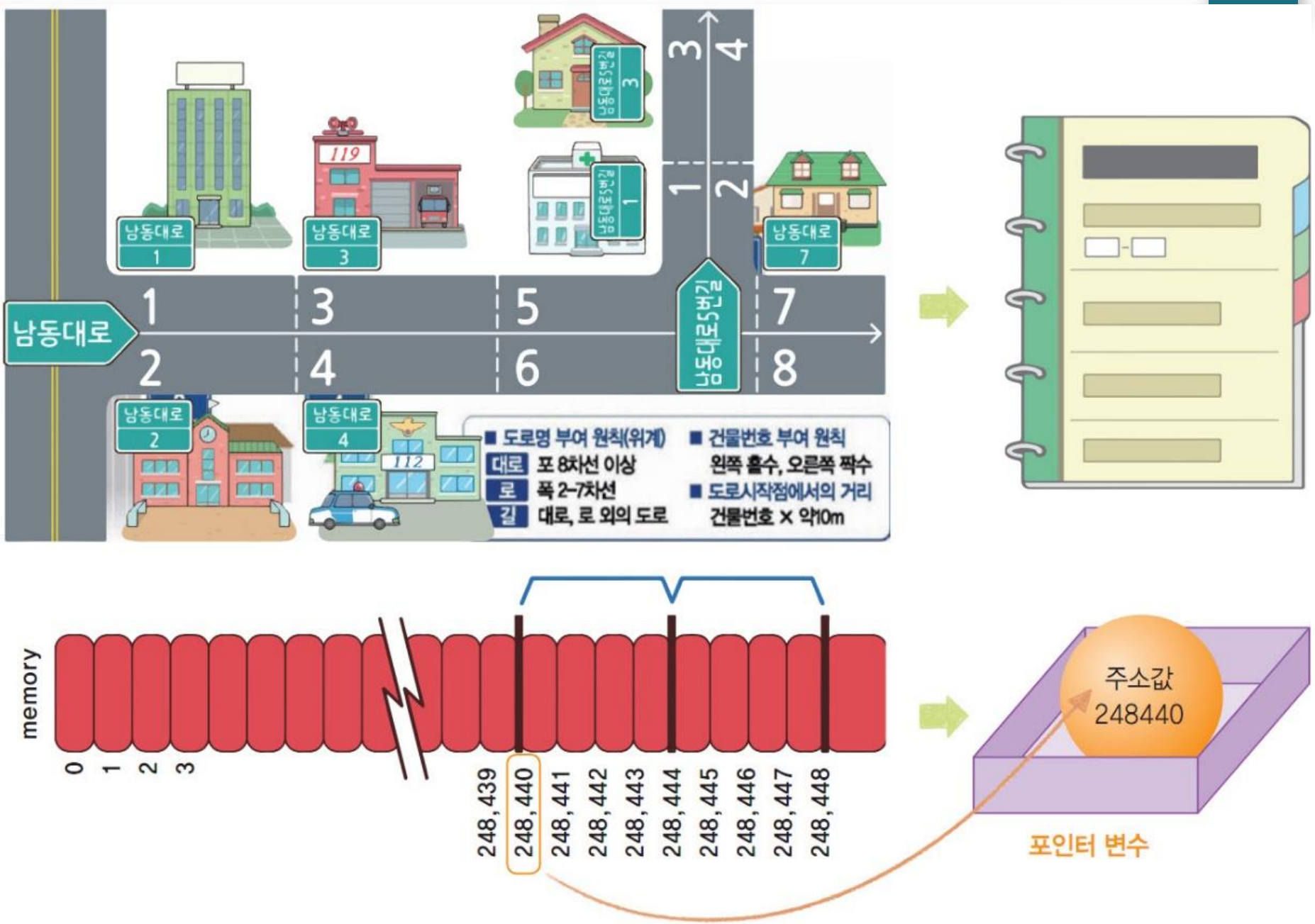


그림 8-3 메모리 주소를 저장하는 포인터 변수

포인터 변수 선언

- **선언 방법**
  - 포인터 변수 선언에서 자료형과 포인터 변수 이름 사이에 연산자 *(asterisk)를 삽입
    - 간단히 포인터라고도 부름

포인터 변수선언

자료형 *변수이름 ;

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

```
int *ptrint; //가장 선호
short*ptrshort;
char * ptrchar;
double *ptrdouble;
```

그림 8-4 포인터 변수선언 구문

  - 예로 'int
    - 'int 포인터 ptrint'라고 읽음
  - 변수 자료형이 다르면
    - 그 변수의 주소를 저장하는 포인터의 자료형도 반드시 달라야 함

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

ptrint = &data;
```

변수 data 는 정수 int를 저장하는 일반변수

변수 ptrint는 정수 int를 저장하는 일반변수의 주소를 저장하는 포인터 변수

포인터 ptrint는 이제 'data를 가리킨다' (data 주소값을 갖는다).

그림 8-5 포인터 변수와 일반 변수의 주소 저장

포인터 선언과 대입(1)

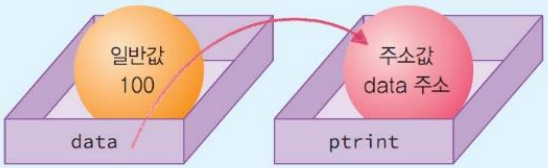
예제 pointer.c

- 변수의 값과 주소 값의 대입과 활용

실습예제 8-2 pointer.c

포인터 변수선언과 주소값 대입

```
01  // file: pointer.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int data = 100;
07     int *ptrint;
08     ptrint = &data;
09
10     printf("변수명   주소값        저장값\n");
11     printf("----------------------------\n");
12     printf("  data   %p   %8d\n", &data, data);
13     printf("ptrint   %p   %p\n", &ptrint, ptrint);
14
15     return 0;
16  }
```



설명

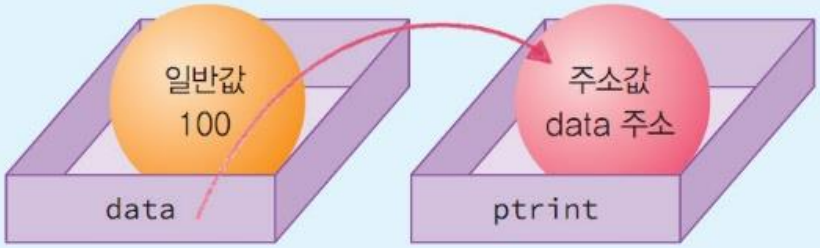
```
06  int 형 변수 data 선언하여 100 저장
07  int 형 포인터 변수 ptrint 선언
08  int 형 포인터 변수 ptrint에 변수 data의 주소를 저장
12  변수 data의 주소(0024FB44)와 내부 저장값 100을 출력
13  변수 ptrint의 주소(0024FB38)와 내부 저장값 0024FB44를 출력, 변수 ptrint의 내부 저장
    값은 바로 data의 주소값인 0024FB44인 것을 확인할 수 있으며, 이러한 주소값은 실행 때마다
    다를 수 있음
```

실행결과

```
변수명      주소값       저장값
----------------------------
  data   0024FB44        100
ptrint   0024FB38   0024FB44
```

실습예제 8-2 pointer.c

포인터 변수선언과 주소값 대입




```c
// file: pointer.c
#include <stdio.h>

int main(void)
{
    int data = 100;
    int *ptrint;
    ptrint = &data;

    printf("변수명  주소값        저장값\n");
    printf("----------------------------\n");
    printf("  data  %p  %8d\n", &data, data);
    printf("ptrint  %p  %p\n", &ptrint, ptrint);

    return 0;
}
```

설명

06 int 형 변수 data 선언하여 100 저장

07 int 형 포인터 변수 ptrint 선언

08 int 형 포인터 변수 ptrint에 변수 data의 주소를 저장

12 변수 data의 주소(0024FB44)와 내부 저장값 100을 출력

13 변수 ptrint의 주소(0024FB38)와 내부 저장값 0024FB44를 출력, 변수 ptrint의 내부 저장값은 바로 data의 주소값인 0024FB44인 것을 확인할 수 있으며, 이러한 주소값은 실행 때마다 다를 수 있음

포인터 선언과 대입(2)

- 포인터 변수도 선언된 후 초기값이 없으면
  - 의미 없는 쓰레기(garbage) 값이 저장
- 문장 ptrint = &data;
  - 포인터 변수 ptrint에 변수 data의 주소를 저장하는 문장
- 포인터 변수 ptrint에서 data 로의 화살표
  - 이러한 관계를 '포인터 변수 ptrint는 변수 data를 가리킨다' 또는 '참조(reference)한다'라고 표현
- 포인터 변수는 가리키는 변수의 크기
  - 종류에 관계없이 크기가 모두 4바이트

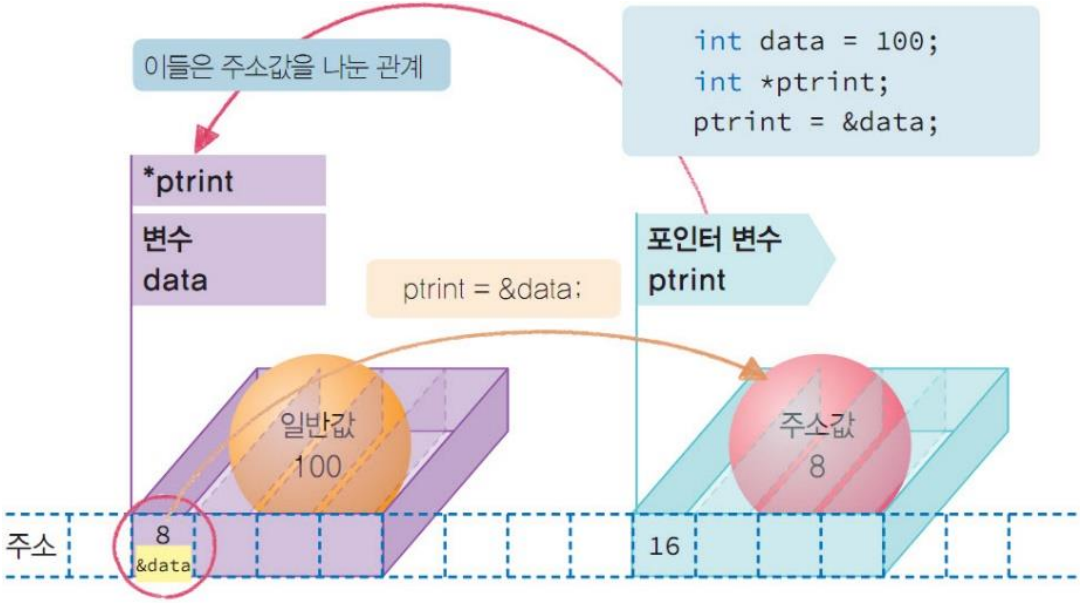
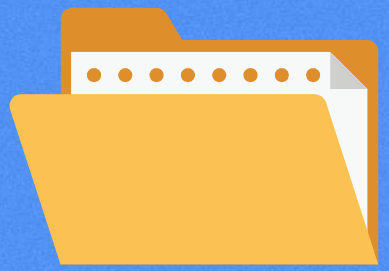


그림 8-6 포인터 변수선언과 주소의 대입



02. 간접 참조 연산자 *와 포인터 연산

여러 포인터 변수선언

- **여러 개의 포인터 변수를 한 번에 선언**
  - 두 번째 선언은 ptr1은 int 형 포인터를 의미하나 ptr2와 ptr3은 단순히 int형

```
int    *ptr1, *ptr2, *ptr3;

int    *ptr1, ptr2, ptr3;
```

- **포인터 변수에 지정할 특별한 초기값이 없는 경우**
  - int *ptr = NULL;
    - 0번 주소값인 NULL로 초기값을 저장
    - NULL이 저장된 포인터 변수는 아무 변수도 가리키고 있지 않다는 의미
- **NULL**
  - 헤더파일 stdio.h에 정의되어 있는 포인터 상수로서 0번지의 주소값을 의미

주소값 NULL

예제 nullpointer.c

- 포인터 변수와 NULL의 활용

- 포인터 변수도 초기값을 저장하도록
  - 아니면 NULL을 저장
- 초기값을 지정하지 않은 포인터 변수 ptr2를 출력
  - "warning C4101: 'ptr2' : 참조되지 않은 지역 변수입니다."라는 컴파일오류가 발생

실습예제 8-3 nullpointer.c

한번에 여러 포인터 변수 선언과 NULL 주소값 대입

```
01  // file: nullpointer.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int *ptr1, *ptr2, data = 10;
07     ptr1 = NULL;
08
09     printf("%p\n", ptr1);
10     //printf("%p\n", ptr2);   <-- 초기값이 없어서 컴파일 오류발생
11     printf("%d\n", data);
12
13     return 0;
14  }
```

설명

```
06  int 형 포인터 변수 ptr1과 ptr2를 선언하면서 int 형 변수 data 선언하여 10 저장
07  int 형 포인터 변수 ptr1에 주소값 0인 NULL을 저장
09  int 형 포인터 변수 ptr1에 저장된 주소값인 0을 출력
10  int 형 포인터 변수 ptr2에 저장된 주소값을 출력하려 하나 저장된 것이 없어 컴파일오류 발생
11  int 형 변수 data에 저장된 값인 10을 출력
```

실행결과

```
00000000
10
```

실습예제 8-3 nullpointer.c

한번에 여러 포인터 변수 선언과 NULL 주소값 대입

```c
// file: nullpointer.c
#include <stdio.h>

int main(void)
{
    int *ptr1, *ptr2, data = 10;
    ptr1 = NULL;

    printf("%p\n", ptr1);
    //printf("%p\n", ptr2);
    printf("%d\n", data);

    return 0;
}
```

초기값이 없어서 컴파일 오류발생

설명

```
int 형 포인터 변수 ptr1과 ptr2를 선언하면서 int 형 변수 data 선언하여 10 저장
int 형 포인터 변수 ptr1에 주소값 0인 NULL을 저장
int 형 포인터 변수 ptr1에 저장된 주소값인 0을 출력
int 형 포인터 변수 ptr2에 저장된 주소값을 출력하려 하나 저장된 것이 없어 컴파일오류 발생
int 형 변수 data에 저장된 값인 10을 출력
```

실행결과

```
00000000
10
```

간접 참조연산자 *

- **간접 참조연산자(indirection operator) *를 사용**
  - 포인터 변수가 갖는 주소로 그 주소의 원래 변수를 참조 가능
  - 포인터 변수가 가리키고 있는 변수를 참조
  - 간접 참조연산자를 이용한 *ptrint
    - 포인터 ptrint가 가리키고 있는 변수 자체를 의미
- **직접참조(direct access)**
  - 변수 data 자체를 사용해 자신을 참조하는 방식
- **간접 참조(indirect access)**
  - *ptrint를 이용해서 변수 data를 참조하는 방식

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

*ptrint = 200;
```

그림 8-8 간접연산자 *와 간접참조

```
int data = 100;
int *ptrint = &data;
char *ptrchar = &ch;
printf("간접참조 출력: %d %c\n", *ptrint, *ptrchar);

*ptrint = 200;
printf("직접참조 출력: %d %c\n", data, ch);
```

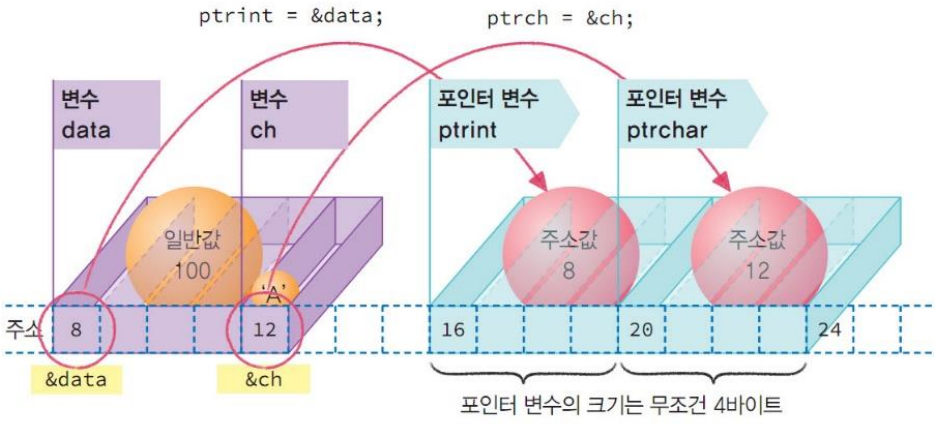


그림 8-9 포인터 변수와 주소값이 저장된 변수 사이의 관계

간접 참조

예제 dereference.c

연산자 &와 *

- 주소 연산자 &와 간접 참조 연산자 *, 모두 전위 연산자로 서로 반대의 역할
  - 주소 연산 '&변수'는 변수의 주소값이 결과값
  - 간접 참조 연산 '*포인터변수'는 포인터 변수가 가리키는 변수 자체가 결과값

실습예제 8-4 dereference.c

포인터 변수와 간접연산자 *를 이용한 간접참조

```
01  // file: dereference.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int data = 100;
07     char ch = 'A';
08     int *ptrint = &data;
09     char *ptrchar = &ch;
10     printf("간접참조 출력: %d %c\n", *ptrint, *ptrchar);
11
12     *ptrint = 200; //변수 data를 *ptrint로 간접참조하여 그 내용을 수정
13     *ptrchar = 'B'; //변수 ch를 *ptrchar로 간접참조하여 그 내용을 수정
14     printf("직접참조 출력: %d %c\n", data, ch);
15
16     return 0;
17  }
```

변수 data와 같다. / 변수 ch와 같다.

설명

```
06  int 형 변수 data 선언하여 100 저장
07  char 형 변수 ch 선언하여 문자 'A' 저장
08  int 형 포인터 변수 ptrint를 선언하여 변수 data의 주소값을 저장
09  char 형 포인터 변수 ptrchar를 선언하여 변수 ch의 주소값을 저장
10  int 형 포인터 변수 ptrint와 간접연산자 *를 이용한 *ptrint를 사용하여 data의 저장값을 출력
10  char 형 포인터 변수 ptrchar와 간접연산자 *를 이용한 *ptrchar를 사용하여 ch의 저장값을
    출력
12  int 형 포인터 변수 ptrint와 간접연산자 *를 이용한 *ptrint를 사용하여 data의 저장값을
    200으로 수정
13  char 형 포인터 변수 ptrchar와 간접연산자 *를 이용한 *ptrchar를 사용하여 ch의 저장값을
    'B'로 수정
14  int 형 변수 data에 저장된 값과 char 형 변수 ch에 저장된 값을 출력
```

실행결과

```
간접참조 출력: 100 A
직접참조 출력: 200 B
```

실습예제 8-4 dereference.c

포인터 변수와 간접연산자 *를 이용한 간접참조

```
01  // file: dereference.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int data = 100;
07     char ch = 'A';
08     int *ptrint = &data;
09     char *ptrchar = &ch;
10     printf("간접참조 출력: %d %c\n", *ptrint, *ptrchar);
11
12     *ptrint = 200; //변수 data를 *ptrint로 간접참조하여 그 내용을 수정
13     *ptrchar = 'B'; //변수 ch를 *ptrchar로 간접참조하여 그 내용을 수정
14     printf("직접참조 출력: %d %c\n", data, ch);
15
16     return 0;
17  }
```

변수 data와 같다.

변수 ch와 같다.

설명

```
06  int 형 변수 data 선언하여 100 저장
07  char 형 변수 ch 선언하여 문자 'A' 저장
08  int 형 포인터 변수 ptrint를 선언하여 변수 data의 주소값을 저장
09  char 형 포인터 변수 ptrchar를 선언하여 변수 ch의 주소값을 저장
10  int 형 포인터 변수 ptrint와 간접연산자 *를 이용한 *ptrint를 사용하여 data의 저장값을 출력
```

주소 연산

- **포인터 변수의 간단한 더하기와 뺄셈 연산**
  - 포인터가 가리키는 변수 크기에 비례한 연산
    - 포인터의 연산은 절대적인 주소의 계산이 아님
  - Int형 포인터 pi에 저장된 주소값이 100이라고 가정
    - (pi+1)은 101이 아니라 주소값 104
    - 즉 (pi+1)은 pi가 가리키는 다음 int형의 주소를 의미

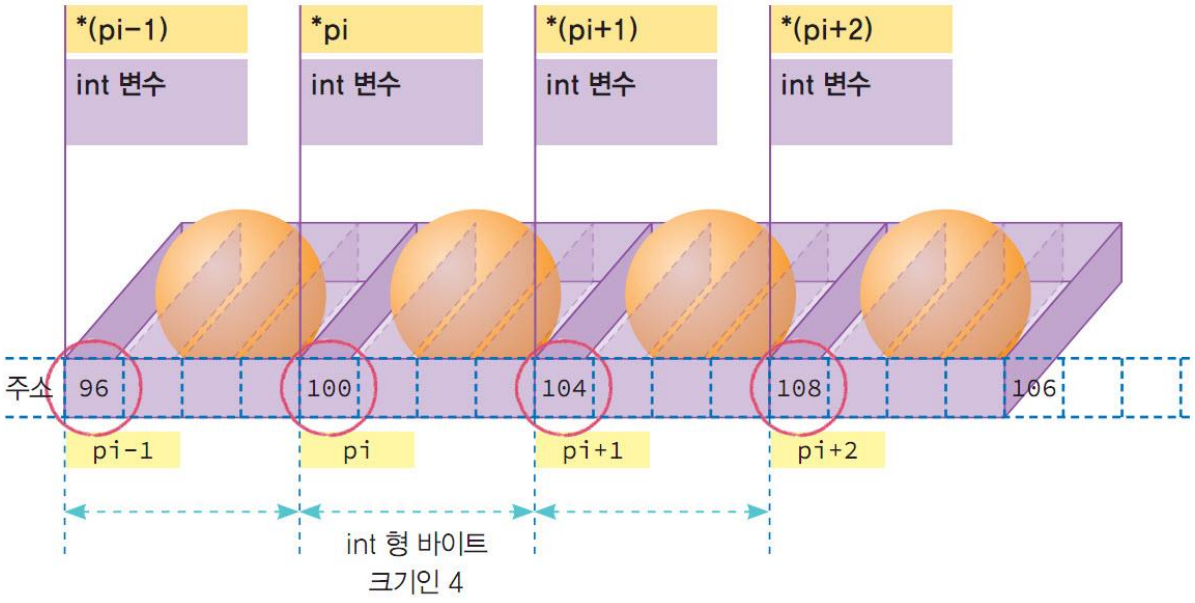


그림 8-10 주소의 연산을 이용한 이웃한 변수의 참조

주소 연산

예제 calcptr.c

연산자 &와 *

- 포인터 pd에 정수 100을 직접 저장하면 경고가 발생
- double *pd = 100; //경고발생
- double *pd = (double *)100; //가능하나 잘 이용하지 않음
  - double형 포인터에 100이라는 주소값을 저장
  - 포인터 자료형으로 100을 변환하는 연산식 (double *) 100을 사용해 저장 가능, 권장하지 않음

실습예제 8-5 calcptr.c

포인터 변수의 간단한 덧셈 뺄셈 연산

```
01  // file: calcptr.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06     char *pc = (char *)100;//가능하나 잘 이용하지 않음
07     int *pi = (int *)100;   //가능하나 잘 이용하지 않음
08     double *pd = (double *)100;//가능하나 잘 이용하지 않음
09     pd = 100;       //경고발생
10
11     printf("%u  %u  %u\n", (int)(pc - 1), (int)pc, (int)(pc + 1));
12     printf("%u  %u  %u\n", (int)(pi - 1), (int)pi, (int)(pi + 1));
13     printf("%u  %u  %u\n", (int)(pd - 1), (int)pd, (int)(pd + 1));
14
15     return 0;
16  }
```

설명

```
06  char형 포인터 변수 pc 선언하여 char형 포인터 주소값 100을 저장
07  int형 포인터 변수 pi 선언하여 int형 포인터 주소값 100을 저장
08  double형 포인터 변수 pd 선언하여 double형 포인터 주소값 100을 저장
09  double형 포인터 변수 pd에 정수 100을 저장하므로 경고 발생
11  char형 포인터 변수 pc에 저장된 값 100과 비교하여 pc-1과 pc+1을 출력, char의 크기가
    1이므로 각각 1만큼의 차이가 나므로 99 100 101 출력
12  int형 포인터 변수 pi에 저장된 값 100과 비교하여 pi-1과 pi+1을 출력, int의 크기가
    4이므로 각각 4만큼의 차이가 나므로 96 100 104 출력
13  double형 포인터 변수 pd에 저장된 값 100과 비교하여 pd-1과 pd+1을 출력, double의 크기가
    8이므로 각각 8만큼의 차이가 나므로 92 100 108 출력
```

실행결과

```
99   100  101
96   100  104
92   100  108
```

실습예제 8-5

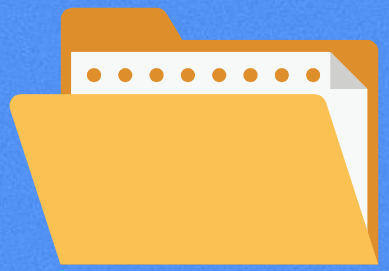
calcptr.c

포인터 변수의 간단한 덧셈 뺄셈 연산

```
01  // file: calcptr.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06      char *pc = (char *)100;//가능하나 잘 이용하지 않음
07      int *pi = (int *)100;    //가능하나 잘 이용하지 않음
08      double *pd = (double *)100;//가능하나 잘 이용하지 않음
09      pd = 100;       //경고발생
10
11      printf("%u  %u  %u\n", (int)(pc - 1), (int)pc, (int)(pc + 1));
12      printf("%u  %u  %u\n", (int)(pi - 1), (int)pi, (int)(pi + 1));
13      printf("%u  %u  %u\n", (int)(pd - 1), (int)pd, (int)(pd + 1));
14
15      return 0;
16  }
```

설명

```
06  char형 포인터 변수 pc 선언하여 char형 포인터 주소값 100을 저장
07  int형 포인터 변수 pi 선언하여 int형 포인터 주소값 100을 저장
08  double형 포인터 변수 pd 선언하여 double형 포인터 주소값 100을 저장
09  double형 포인터 변수 pd에 정수 100을 저장하므로 경고 발생
```



03. 포인터 형변환

내부 저장 표현

- **변수 value에 16진수 0x61626364를 저장**
  - 만일 변수 value의 주소가 56번지라면
    - 56번지에는 16진수 64가 저장
    - 다음 주소 57번지에는 63이 저장
    - 다음에 각각 62, 61이 저장

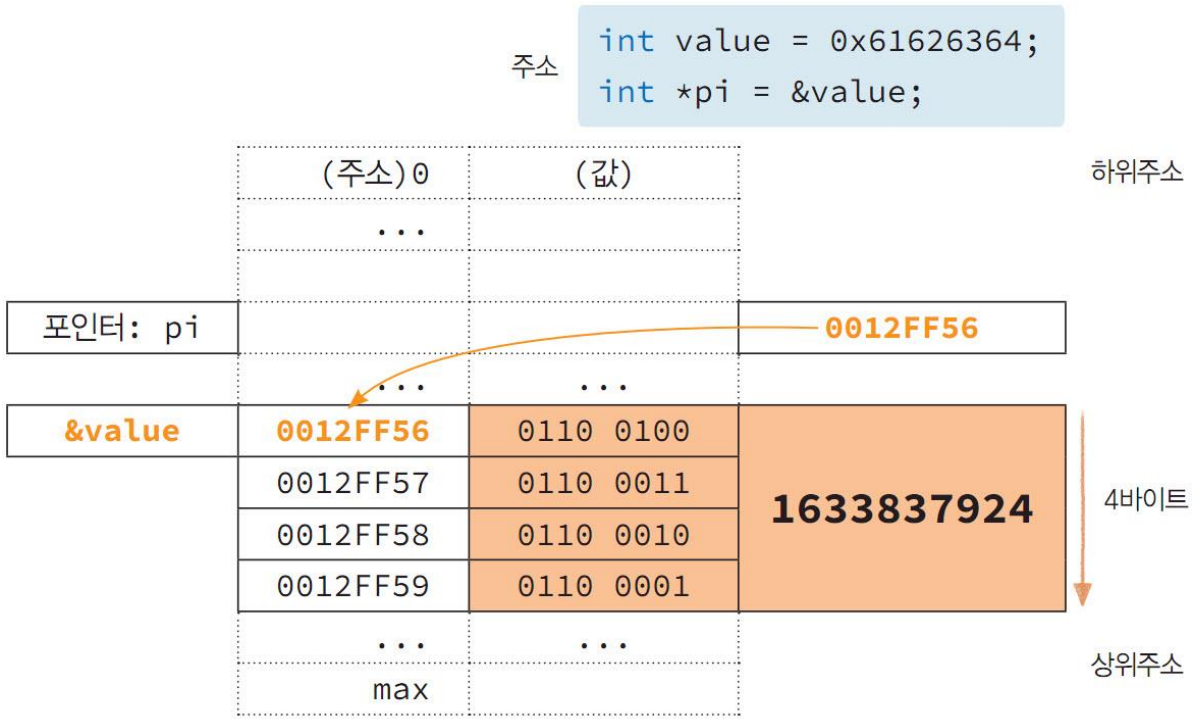


그림 8-12 지역변수가 선언된 메모리 내부 모습

명시적 포인터 형변환

예제 ptrtypecast.c

- 정수의 내부를 각각 1바이트 씩 문자로 출력

포인터 형변환

- 포인터 변수는 동일한 자료형끼리만 대입이 가능
  - 포인터의 자료형이 다르면 경고가 발생
  - 필요하면 명시적으로 형변환을 수행 가능

실습예제 8-7 ptrtypecast.c

포인터 자료형의 변환

```
01  // file: ptrtypecast.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int value = 0x61626364;
07     int *pi = &value;
08     char *pc = (char *) &value;
09
10     printf("변수명    저장값       주소값\n");
11     printf("----------------------------\n");
12     printf(" value   %0#x  %u\n", value, pi); //정수 출력
13
14     //문자 포인터로 문자 출력 모듈
15     for (int i = 0; i <= 3; i++)
16     {
17        char ch = *(pc + i);
18     printf("*(pc+%d)  %0#6x  %2c  %u\n", i, ch, ch, pc + i);
19     }
20
21     return 0;
22  }
```

문자를 6자리의 16진수로 출력

문자를 직접 출력

설명

```
06  int 형 변수 value를 선언하여 16진수 0x61626364을 저장, 0x61은 1바이트인 문자로 해석하면
    문자 'a'이며, 마찬가지로 0x62는 문자 'b'이며, 0x63은 문자 'c'이고, 0x64은 문자 'd'임
07  int 포인터 pi 선언하여 int 변수 value의 주소를 저장
08  char 포인터 pc 선언하여 int 변수 value의 주소를 char 포인터로 형변환하여 저장, 이제 pc는
    char 포인터이므로 1바이트씩 이동 가능
10  출력을 위한 제목 헤드라인 출력
11  출력을 위한 헤드라인 출력
12  변수 value의 변수명 저장값 주소값 출력, 저장값은 16진수로 출력
15  반복 for는 제어변수 i가 0, 1, 2, 3까지 4회 실행
16~19 반복몸체가 두 문장이므로 반드시 블록이 필요
17  char 변수 ch에 (pc+i)가 가리키는 문자 변수를 저장
18  (pc+i)가 가리키는 문자 변수의 변수명 저장값 주소값 출력, 저장값은 16진수와 문자 각각 2번 출력
```

실행결과

```
변수명      저장값        주소값
----------------------------
 value   0x61626364  0045F910
*(pc+0)  0x0064   d  0045F910
*(pc+1)  0x0063   c  0045F911
*(pc+2)  0x0062   b  0045F912
*(pc+3)  0x0061   a  0045F913
```

ptrtypecast.c

포인터 자료형의 변환

```c
// file: ptrtypecast.c
#include <stdio.h>

int main(void)
{
    int value = 0x61626364;
    int *pi = &value;
    char *pc = (char *) &value;

    printf("변수명    저장값       주소값\n");
    printf("-------------------------\n");
    printf(" value    %0#x  %u\n", value, pi); //정수 출력

    //문자 포인터로 문자 출력 모듈
    for (int i = 0; i <= 3; i++)
    {
        char ch = *(pc + i);
    printf("*(pc+%d)  %0#6x  %2c  %u\n", i, ch, ch, pc + i);
    }
    return 0;
}
```

u ← p

u ← p

문자를 6자리의 16진수로 출력

문자를 직접 출력