C 언어 EXPRESS(개정3판)

CH08 포인터

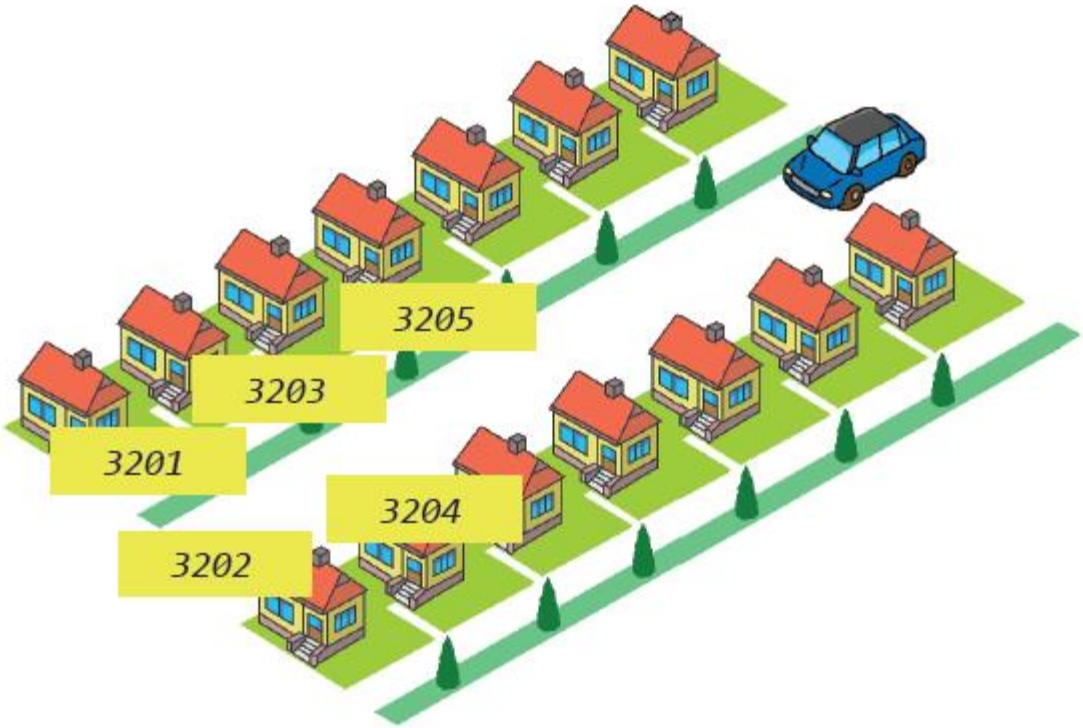
이번 장에서 학습할 내용

- 포인터이란?
- 변수의 주소
- 포인터의 선언
- 간접 참조 연산자
- 포인터 연산
- 포인터와 배열
- 포인터와 함수

© 2021 생능출판사 All rights reserved

포인터란?

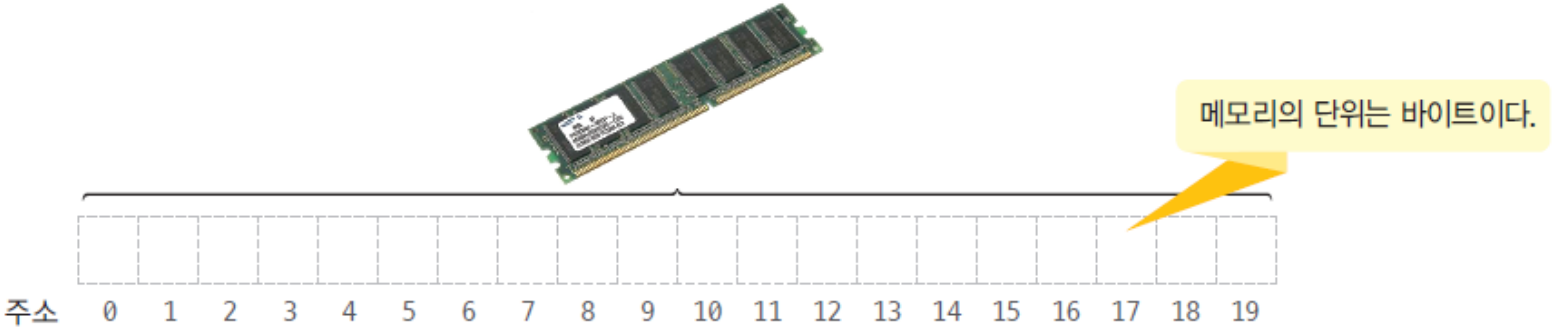
- *포인터(pointer):* 주소를 가지고 있는 변수



© 2021 생능출판사 All rights reserved

변수에 어디에 저장되는가?

- 변수는 메모리에 저장된다.
- 메모리는 바이트 단위로 액세스된다.
  - 첫번째 바이트의 주소는 0, 두번째 바이트는 1,...

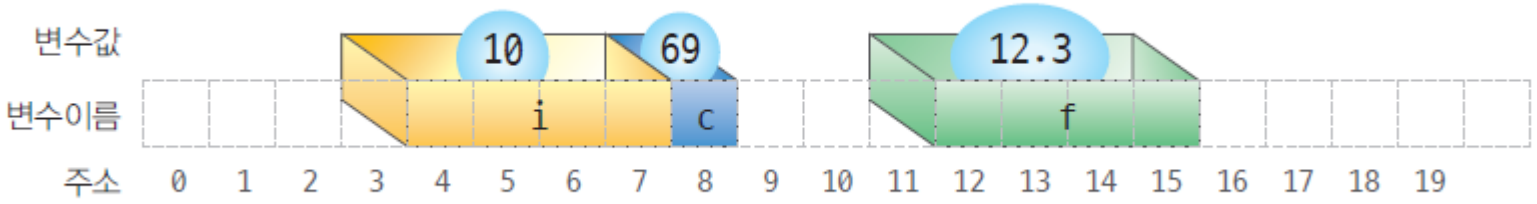


© 2021 생능출판사 All rights reserved

변수와 메모리

- 변수의 크기에 따라서 차지하는 메모리 공간이 달라진다.
- char형 변수: 1바이트, int형 변수: 4바이트,...

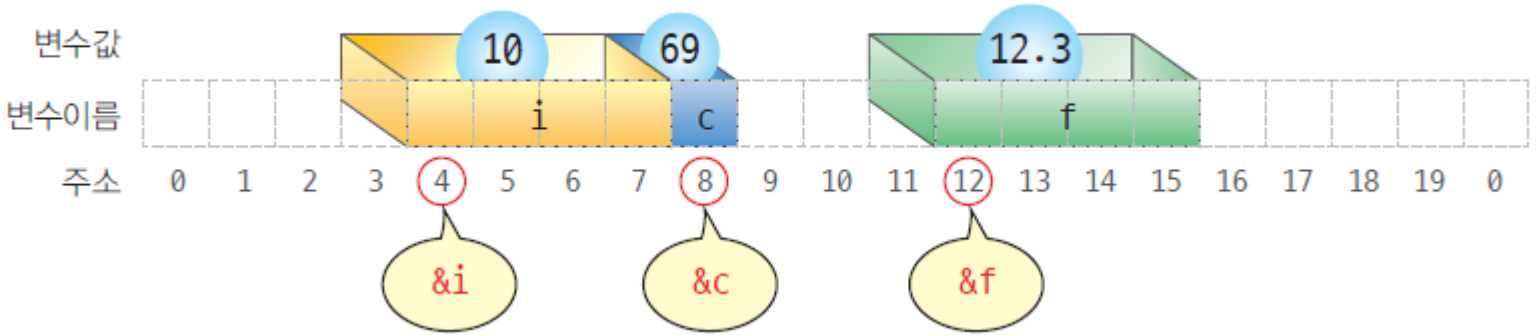
```
int main(void)
{
  int i = 10;
  char c = 69;
  float f = 12.3;
}
```



© 2021 생능출판사 All rights reserved

변수의 주소

- 변수의 주소를 계산하는 연산자: &
- 변수 i의 주소: &i



© 2021 생능출판사 All rights reserved

주의

- 여러 개의 포인터 변수를 한 줄에 선언할 때는 주의하여야 한다. 다음과 같이 선언하는 것은 잘못되었다.
  - int *p1, p2, p3; // (×) p2와 p3는 정수형 변수가 된다.

- 올바르게 선언하려면 다음과 같이 하여야 한다.
  - int *p1, *p2, *p3; // (○) p2와 p3는 정수형 변수가 된다.

© 2021 생능출판사 All rights reserved

포인터의 선언

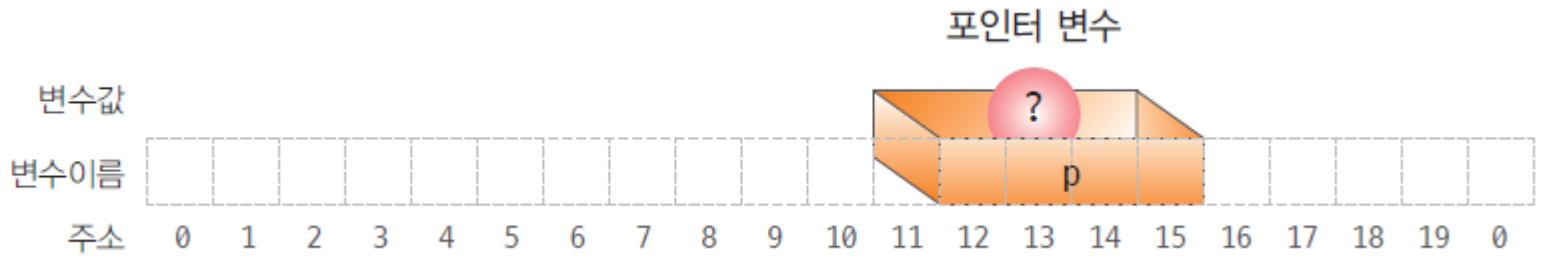
- 포인터: 변수의 주소를 가지고 있는 변수

Syntax: 포인터 선언

예

```
int * p;
```

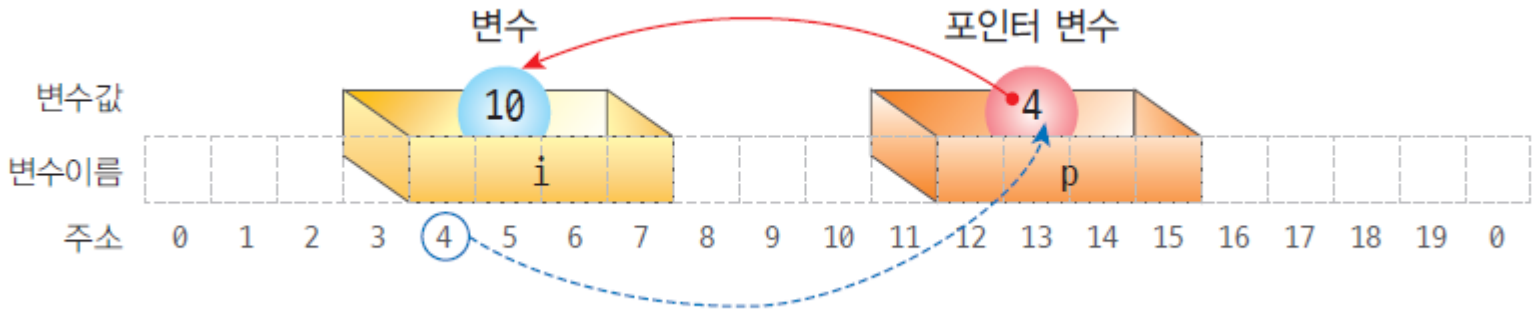
정수를 가리키는 포인터 p



© 2021 생능출판사 All rights reserved

포인터와 변수의 연결

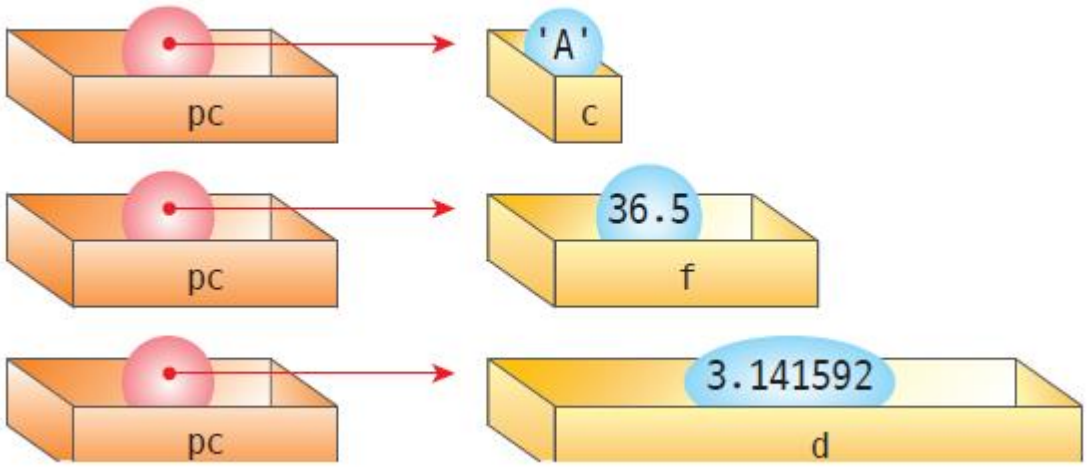
```
int i = 10;          // 정수형 변수 i 선언
int *p;              // 포인터 변수 p 선언
p = &i;              // 변수 i의 주소가 포인터 p로 대입
```



© 2021 생능출판사 All rights reserved

다양한 포인터의 선언

```
char c = 'A';                   // 문자형 변수 c
float f = 36.5;                 // 실수형 변수 f
double d = 3.141592;            // 실수형 변수 d

char *pc = &c;                  // 문자를 가리키는 포인터 pc
float *pf = &f;                 // 실수를 가리키는 포인터 pf
double *pd = &d;                // 실수를 가리키는 포인터 pd
```



© 2021 생능출판사 All rights reserved

포인터 사용시 주의점

- 포인터의 타입과 변수의 타입은 일치하여야 한다.

```
int i;
double *pd;

pd = &i;          // 오류!
*pd = 36.5;
```

- 포인터가 아무것도 가리키고 있지 않는 경우에는 NULL로 초기화
- NULL 포인터를 가지고 간접 참조하면 하드웨어로 감지할 수 있다.

```
int *p = NULL;
```

© 2021 생능출판사 All rights reserved

예제

ch08_ex1.c

```
#include <stdio.h>

int main(void)
{

    int i = 10;
    double f = 12.3;
    int* pi = NULL;
    double* pf = NULL;

    pi = &i;
    pf = &f;

    printf("%p %p\n", pi, &i);
    printf("%p %p\n", pf, &f);
    return 0;
}
```

변수에 할당 되는 주소 값은 실행될때마다 변경됨
따라서 아래 주소값은 다른값이 출력될 수 있음

```
1768820 1768820
1768804 1768804
```

© 2021 생능출판사 All rights reserved

참고

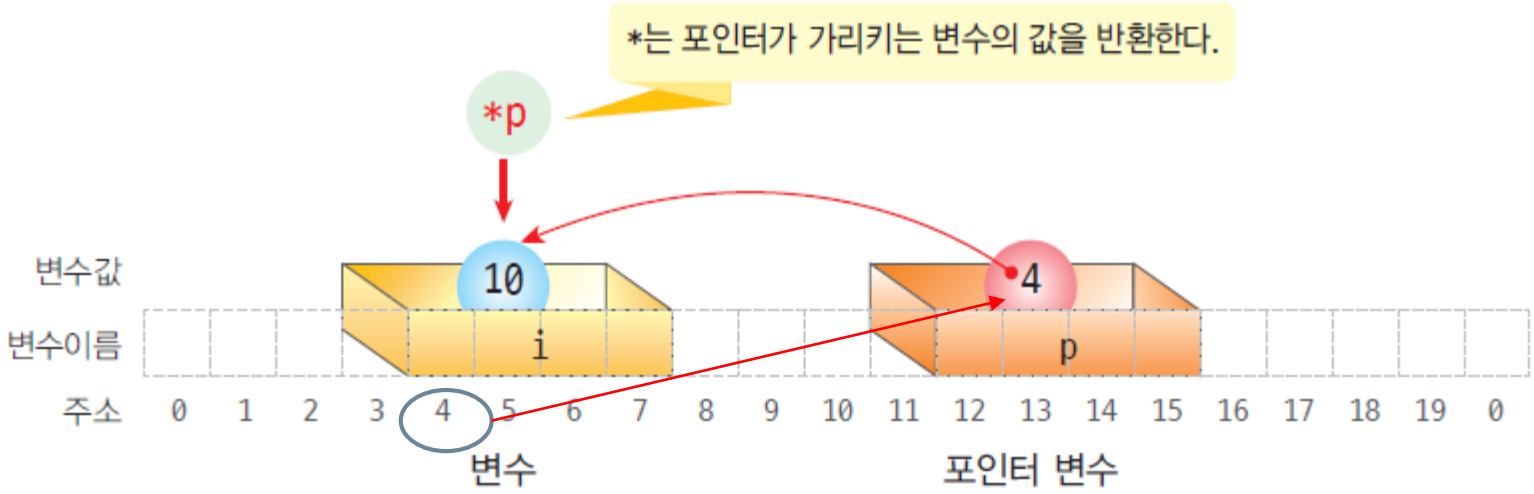
- NULL은 stdio.h 헤더 파일에 다음과 같이 정의된 포인터 상수로 0번지를 의미한다.
  - #define NULL ((void *)0)
- 0번지는 일반적으로는 사용할 수 없다(CPU가 인터럽트를 위하여 사용한다). 따라서 포인터 변수의 값이 0이면 아무 것도 가리키고 있지 않다고 판단할 수 있다.

© 2021 생능출판사 All rights reserved

간접 참조 연산자

- 간접 참조 연산자 *: 포인터가 가리키는 값을 가져오는 연산자

```
int i = 10;

int* p;
p = &i;

printf("%d \n", *p);
```

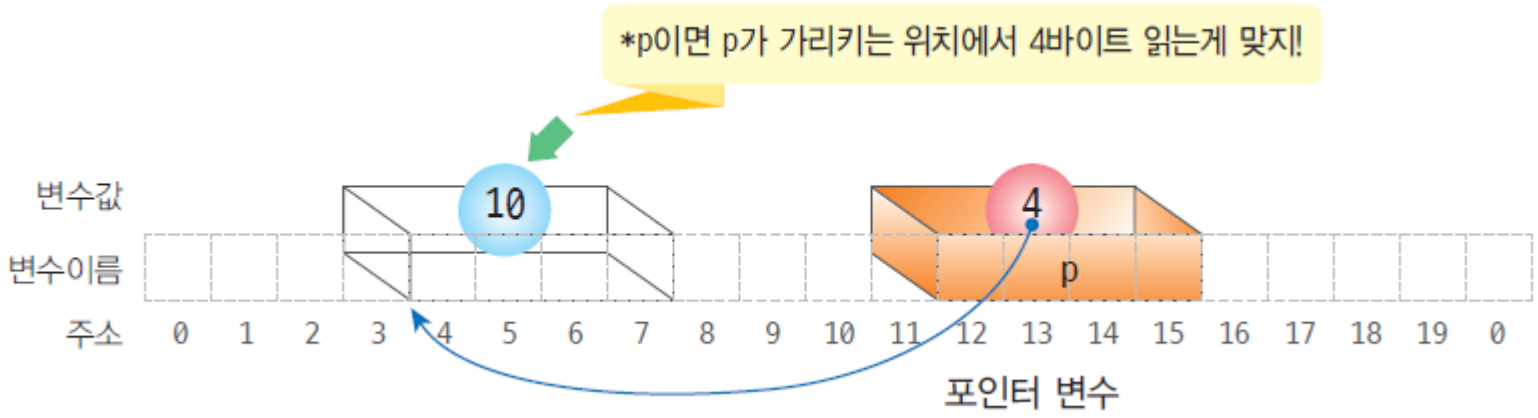


© 2021 생능출판사 All rights reserved

간접 참조 연산자의 해석

- 간접 참조 연산자: 지정된 위치에서 포인터의 타입에 따라 값을 읽어 들인다.

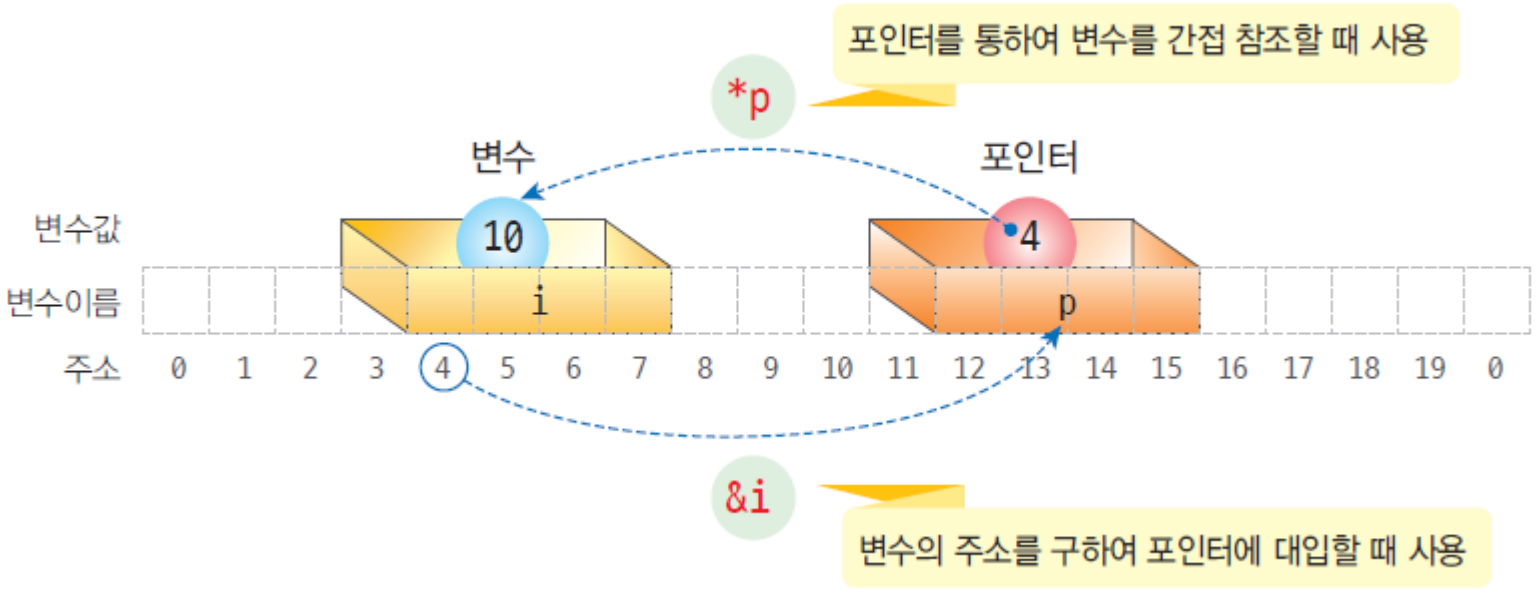
```
int *pi = (int *)10000;
char *pc = (char *)10000;
double *pd = (double *)10000;
```



© 2021 생능출판사 All rights reserved

& 연산자와 * 연산자

- & 연산자: 변수의 주소를 반환한다
- * 연산자: 포인터가 가리키는 곳의 내용을 반환한다.



© 2021 생능출판사 All rights reserved

포인터 예제

ch08_ex2.c

```c
#include <stdio.h>

int main(void)
{
   int i = 10;
   int j = 20;
   int* p = NULL;

   printf("i = %d\n", i);// 변수의 값 출력
   printf("&i = %p\n\n", &i);// 변수의 주소 출력

   p = &i;
   printf("p = 0%p\n", p);// 포인터의 값 출력
   printf("*p = %d\n", *p);// 포인터를 통한 간접 참조 값 출력

   p = &j;
   printf("p = 0%p\n", p);// 포인터의 값 출력
   printf("*p = %d\n", *p);// 포인터를 통한 간접 참조 값 출력

   return 0;
}
```

```
i = 10
&i = 000000B8C913FC14

p = 0000000B8C913FC14
*p = 10
p = 0000000B8C913FC34
*p = 20
```

© 2021 생능출판사 All rights reserved

포인터 예제

ch08_ex3.c

```c
#include <stdio.h>
int main(void)
{
    int i=10;
    int *p;

    p = &i;
    printf("i = %d\n", i);

    *p = 20;
    printf("i = %d\n", i);
    return 0;
}
```

포인터를 통하여 변수의 값을 변경한다.

```
i = 10
i = 20
```

© 2021 생능출판사 All rights reserved

인수 전달 방법

- 함수 호출 시에 인수 전달 방법
  - 값에 의한 호출(call by value)
    - 함수로 복사본이 전달된다.
    - C언어에서의 기본적인 방법
  - 참조에 의한 호출(call by reference)
    - 함수로 원본이 전달된다.
    - C에서는 포인터를 이용하여 흉내 낼 수 있다.

© 2021 생능출판사 All rights reserved

인수 전달 방법

ch08_ex4.c

```
#include <stdio.h>

void callByValue(int x) {
    x = 15;
    printf("Inside : x = %d\n", x);
}

int main() {
    int i = 5;
    printf("Before call: i = %d\n", i);
    callByValue(i);
    printf("After call: i = %d\n", i);
    return 0;
}
```

```
Before call: i = 5
Inside : x = 15
After call: i = 5
```

ch08_ex5.c

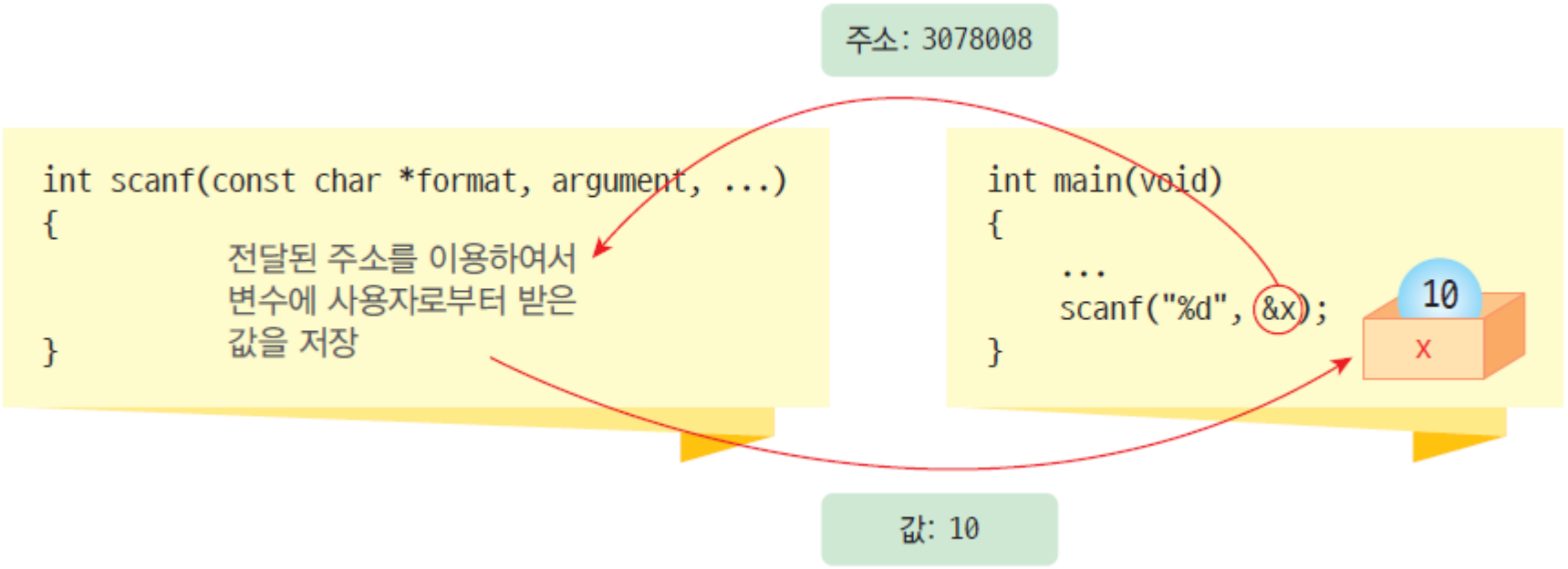
```
#include <stdio.h>

void callByReference(int* x) {
    *x = 15;
    printf("Inside : x = %d\n", *x);
}

int main() {
    int i = 5;
    printf("Before call: i = %d\n", i);
    callByReference(&i);
    printf("After call: i = %d\n", i);
    return 0;
}
```

```
Before call: i = 5
Inside : x = 15
After call: i = 15
```

© 2021 생능출판사 All rights reserved

scanf() 함수

- 변수에 값을 저장하기 위하여 변수의 주소를 받는다.

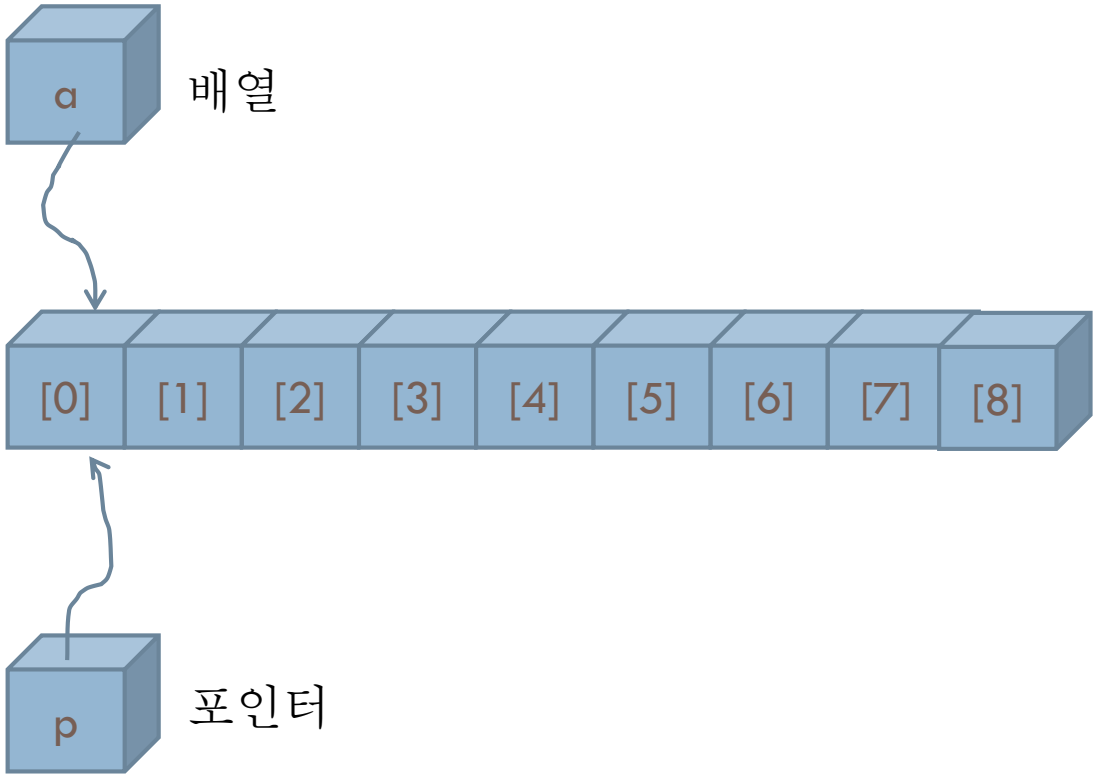


© 2021 생능출판사 All rights reserved

포인터와 배열

- 배열과 포인터는 아주 밀접한 관계를 가지고 있다.
- 배열 이름이 바로 포인터이다.
- 포인터는 배열처럼 사용이 가능하다.

배열 포인터

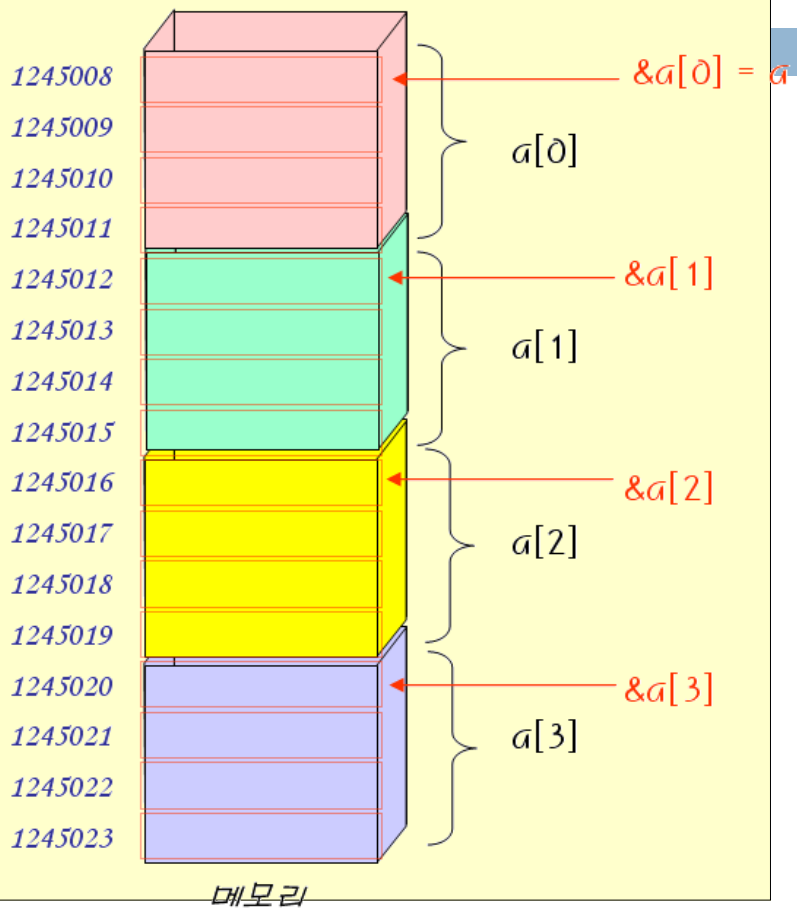


© 2021 생능출판사 All rights reserved

포인터와 배열

ch08_ex06.c

```c
// 포인터와 배열의 관계
#include <stdio.h>

int main(void)
{
    int a[] = { 10, 20, 30, 40, 50 };

    printf("&a[0] = %p\n", &a[0]);
    printf("&a[1] = %p\n", &a[1]);
    printf("&a[2] = %p\n", &a[2]);

    printf("a = %p\n", a);

    return 0;
}
```




```
&a[0] = 000000F50F57F718
&a[1] = 000000F50F57F71C
&a[2] = 000000F50F57F720
a = 000000F50F57F718
```

© 2021 생능출판사 All rights reserved

예제

ch08_ex07.c

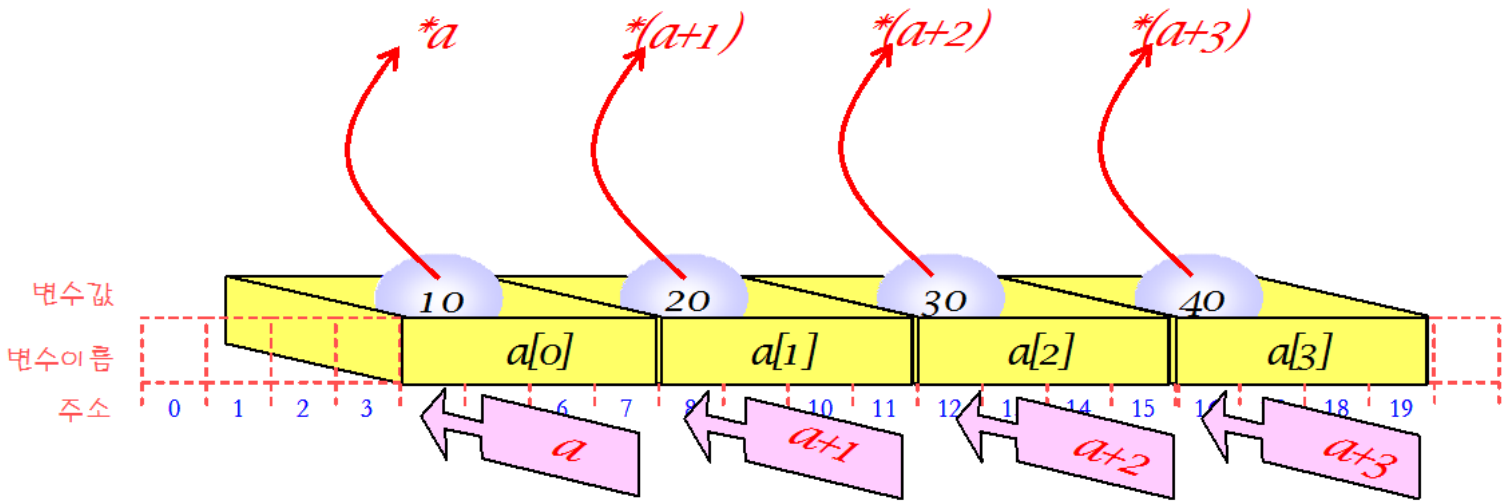
```
// 포인터와 배열의 관계
#include <stdio.h>

int main(void)
{
    int arr[] = { 10, 20, 30, 40, 50 };

    printf("arr = %p\n", arr);
    printf("*arr = %d\n", *arr);
    printf("*(arr+1) = %d\n", *(arr + 1));

    int* p = NULL;
    p = arr;
    printf("p = %p\n", p);
    printf("*p = %d\n", *p);
    printf("*(p+1) = %d\n", *(p + 1));
    return 0;
}
```

```
arr = 0000001BA34FF9F8
*arr = 10
*(arr+1) = 20
p = 0000001BA34FF9F8
*p = 10
*(p+1) = 20
```

© 2021 생능출판사 All rights reserved

포인터와 배열

- 포인터는 배열처럼 사용할 수 있다.
- 인덱스 표기법을 포인터에 사용할 수 있다.



© 2021 생능출판사 All rights reserved

배열 매개 변수

- 일반 매개 변수 vs 배열 매개 변수

```
// 매개 변수 x에 기억 장소가 할당
void sub(int x)
{
    ...
}
```

```
// b에 기억 장소가 할당되지 않는다.
void sub( int b[] )
{
    ...
}
```

- Why? -> 배열을 함수로 복사하려면 많은 시간 소모

© 2021 생능출판사 All rights reserved

배열 매개 변수

- 배열 매개 변수는 포인터로 생각할 수 있다.

```
int main(void)
{
        int a[3]={ 1, 2, 3 };

        sub(a, 3);
}
```

배열의 이름은 포인터이다.

```
void sub(int b[], int size)
{
        *b = 4;
        *(b+1) = 5;
        *(b+2) = 6;
}
```

포인터 b를 통하여 원본 배열을 변경할 수 있다.

© 2021 생능출판사 All rights reserved

ch08_ex08.c

```
// 포인터와 함수의 관계
#include <stdio.h>

void sub(int p_arr[], int n);

int main(void)
{
    int arr[3] = { 1,2,3 };

    printf("%d %d %d\n", arr[0], arr[1], arr[2]);
    sub(arr, 3);
    printf("%d %d %d\n", arr[0], arr[1], arr[2]);

    return 0;
}

void sub(int p_arr[], int n)
{
    p_arr[0] = 4;
    p_arr[1] = 5;
    p_arr[2] = 6;
}
```

```
1 2 3
4 5 6
```

© 2021 생능출판사 All rights reserved

다음 2가지 방법은 완전히 동일하다.

```
// 배열 매개 변수
void sub(int b[], int size)
{
        b[0] = 4;
        b[1] = 5;
        b[2] = 6;
}
```

배열의 이름과 포인터는 근본적으로 같다.

```
// 포인터 매개 변수
void sub(int *b, int size)
{
        b[0] = 4;
        b[1] = 5;
        b[2] = 6;
}
```

배열 표기법을 사용하여 배열에 접근

© 2021 생능출판사 All rights reserved