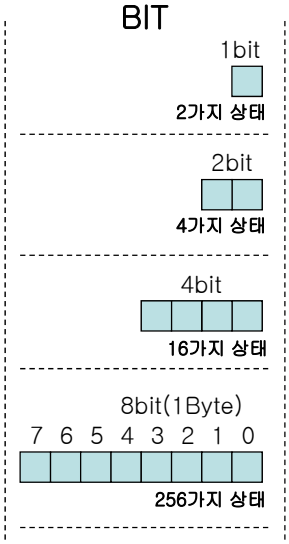
1.1 변수와 메모리

- 목차
  - 메모리
  - 자료형
  - char형 변수
  - int형 변수
  - double형 변수

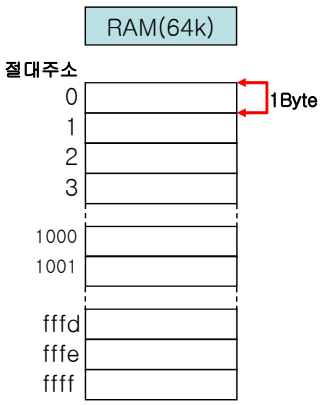
- 메모리

10진수	16진수
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10 →	a
11 →	b
12 →	c
13 →	d
14 →	e
15 →	f
16	10

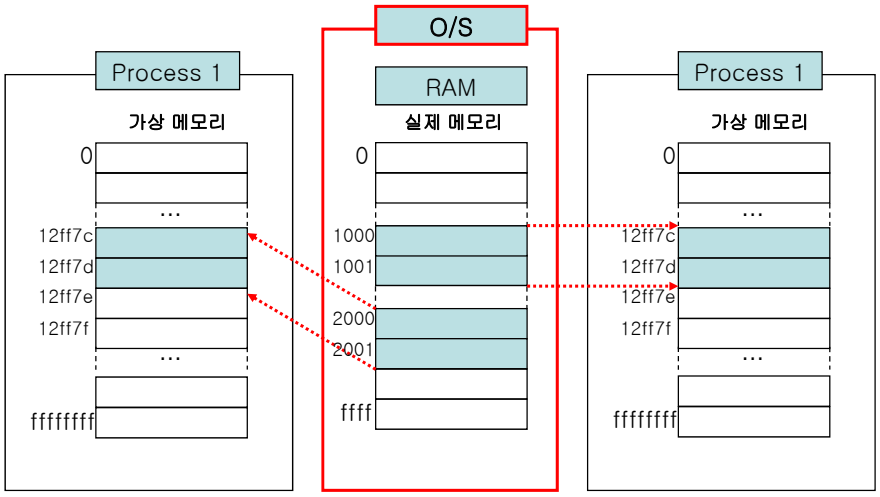
BIT



기억장치 메모리



- 메모리



3 / 123

- 자료형

- 변수 선언을 위한 형태 제공

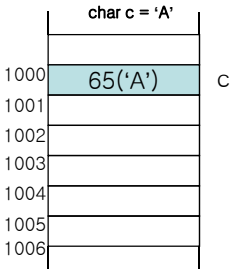
	자료형	메모리 크기	데이터의 범위
정수형	char	1Byte	-128 ~ +127
	short	2Byte	-32768 ~ +32767
	int	4Byte	-214743648 ~ +2147483647
	long	4Byte	-214743648 ~ +2147483647
실수형	float	4Byte	$\pm(3.4*10^{-37} \sim 3.4*10^{+38})$
	double	8Byte	$\pm(1.7*10^{-307} \sim 1.7*10^{+308})$

4 / 123

- char형 변수

- 변수는 메모리 공간의 이름

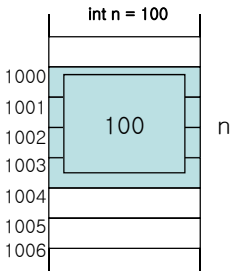
```
#include <stdio.h>
void main ( )
{
      char   c = 'A';


      printf("%d₩n", sizeof(c));
      printf("%c₩n", c);
}
```



5 / 123

- int형 변수

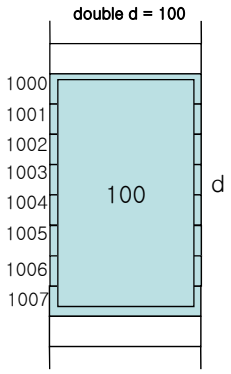
```
#include <stdio.h>
void main ( )
{
      int   n = 100;


      printf("%d₩n", sizeof(n));
      printf("%d₩n", n);
}
```



6 / 123

- double형 변수

```
#include <stdio.h>
void main ( )
{
      double   d = 100;

      printf("%d₩n", sizeof(d));
      printf("%g₩n", d);
}
```



자료형의 주소 = 정수 * sizeof(자료형)

1.2 변수의 주소

- 목차
  - 변수 주소의 의미
  - char형 주소
  - char형 주소의 연산
  - int형 주소
  - int형 주소의 연산
  - 다른 자료형의 주소

- 변수 주소의 의미

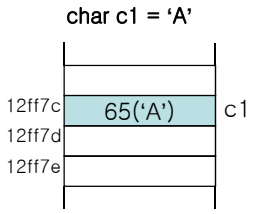
- 변수이름에 '&'연산자를 붙이면 변수의 주소를 의미

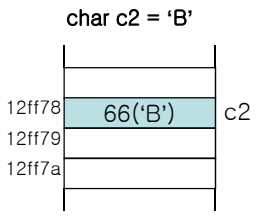
```
#include <stdio.h>
void main ( )
{
      char     c;
      int      n;
      double  d;

      printf( "%x   %x   %x₩n",  &c,  &n,  &d );
}
```

9 / 123

- char형 주소

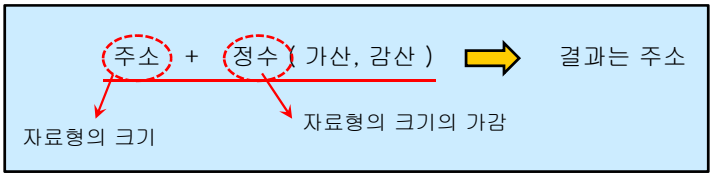
```
#include <stdio.h>
void main ( )
{
      char   c1 = 'A';
      char   c2 = 'B';


      printf("%x₩n", &c1 );
      printf("%x₩n", &c2 );
}
```



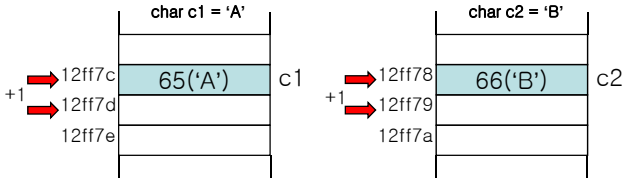


10 / 123

- char형 주소의 연산

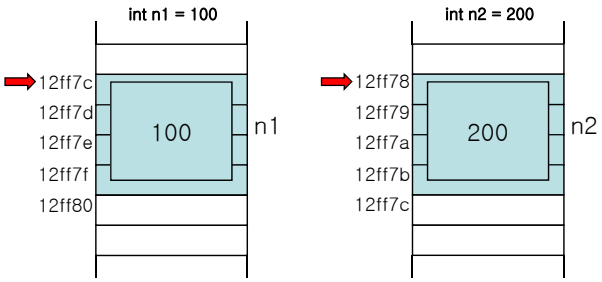



```
#include <stdio.h>
void main ( )
{
      char   c1 = 'A';
      char   c2 = 'B';


      printf("%x₩n", &c1 + 1);
      printf("%x₩n", &c2 + 1);
}
```



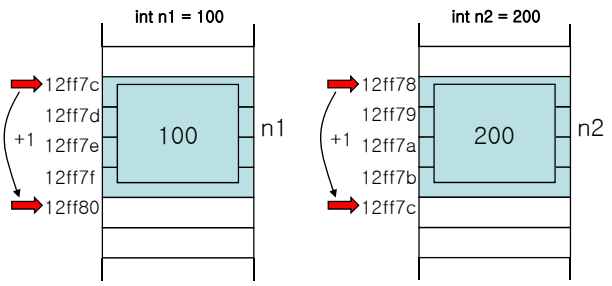
- int형 주소

```
#include <stdio.h>
void main ( )
{
      int   n1 = 100;
      int   n2 = 200;


      printf("%x₩n", &n1);
      printf("%x₩n", &n2);
}
```



- int형 주소의 연산

```
#include <stdio.h>
void main ( )
{
      int   n1 = 100;
      int   n2 = 200;

      printf("%x₩n",  &n1 + 1 );
      printf("%x₩n",  &n2 + 1 );
}
```



13 / 123

1.3 주소의 가감산

- 목차
  - 형변환
  - char형 주소를 int형 주소로
  - int형 주소를 char형 주소로

14 / 123

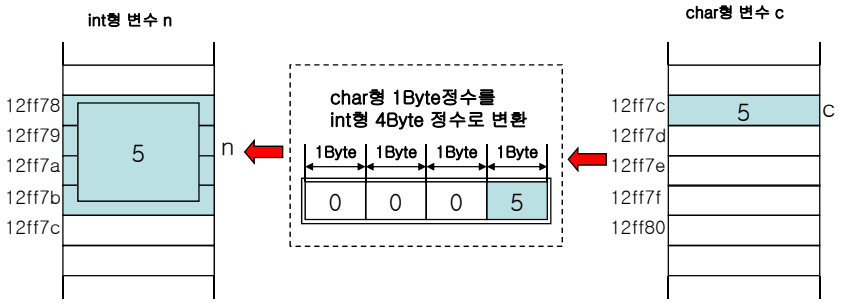
– 형변환

- ( ) 연산자를 사용한 형변환

```
#include <stdio.h>
void main ( )
{
      char    c = 5;
      int       n;


      n = ( int ) c;
      printf("%d₩n", n);
}
```

15 / 123

– 형변환



16 / 123

- 형변환

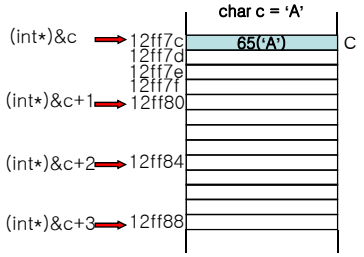
char : char형 자료형으로 변환하라는 의미
char* : char형 주소로 자료형을 변환하라는 의미

```
#include <stdio.h>
void main ( )
{
      char   c = 'A';
      int      n = 20;

      printf("%x %x₩n", &c,  (int*) &c );
      printf("%x %x₩n", &n,  (char*) &n );
}
```

17 / 123

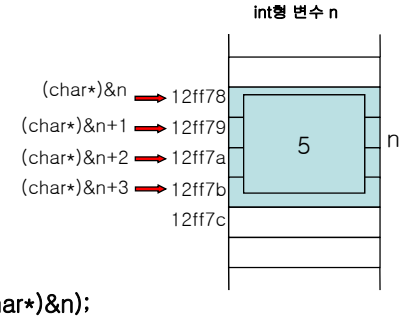
- char형 주소를 int형 주소로

```
#include <stdio.h>
void main( )
{
      char c = 'A';

      printf("%x %x %x₩n", &c,  (char*)&c,    (int*)&c); // 12ff7c, 12ff7c,12ff7c
      printf("%x %x %x₩n", &c + 1, (char*)&c + 1, (int*)&c + 1);
      printf("%x %x %x₩n", &c + 2, (char*)&c + 2, (int*)&c + 2);
      printf("%x %x %x₩n", &c + 3, (char*)&c + 3, (int*)&c + 3);
}
```



18 / 123

- int형 주소를 char형 주소로




```
#include <stdio.h>
void main( )
{
      int   n = 10;

      printf("%x %x ₩n", &n,       (char*)&n);
      printf("%x %x ₩n", &n + 1, (char*)&n + 1);
      printf("%x %x ₩n", &n + 2, (char*)&n + 2);
      printf("%x %x ₩n", &n + 3, (char*)&n + 3);
}
```

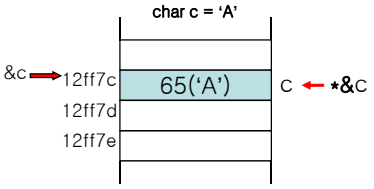

1.4 주소와 메모리

- 목차
  - char형 주소의 메모리접근
  - int형 주소의 메모리접근
  - 주소형 변환 후 메모리접근
  - [ ]연산자를 이용한 메모리접근

- char형 주소의 메모리접근

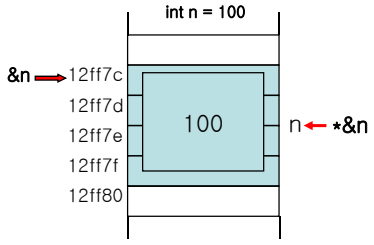
- * 연산자를 이용한 메모리 접근

```
#include <stdio.h>
void main( )
{
      char   c = 'A';

      printf("%x₩n", &c);
      printf("%c %c₩n", c, *&c);
}
```



- int형 주소의 메모리접근

```
#include <stdio.h>
void main ( )
{
      int   n = 100;

      printf("%x₩n", &n);
      printf("%d %d₩n", n, *&n);
}
```

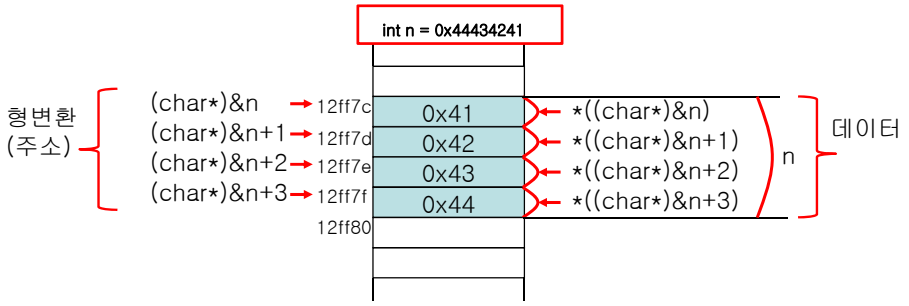


- 주소형 변환 후 메모리접근

```
#include   <stdio.h>
void main ( )
{
      int   n = 0x44434241;


      printf("%d %c₩n", *(char*)&n,      *(char*)&n);      // 65  A
      printf("%d %c₩n", *((char*)&n+1), *((char*)&n+1)); // 66  B
      printf("%d %c₩n", *((char*)&n+2), *((char*)&n+2)); // 67  C
      printf("%d %c₩n", *((char*)&n+3), *((char*)&n+3)); // 68  D
}
```

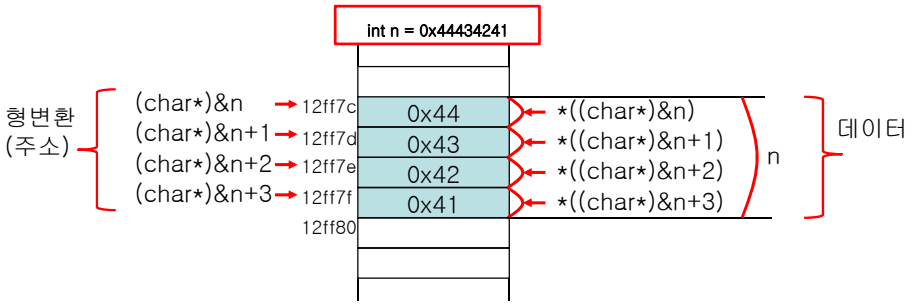
- 주소형 변환 후 메모리접근

Little - endian : 작은단위가 앞에 나오는 방식

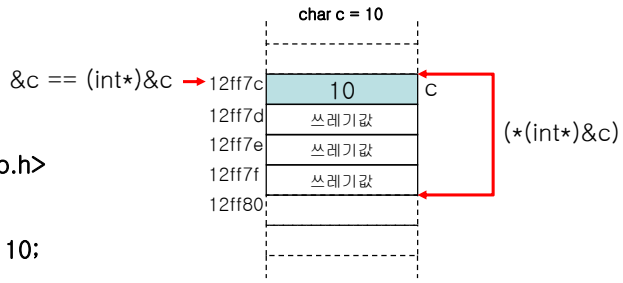


- 주소형 변환 후 메모리접근

Big - endian : 큰 단위가 앞에 나오는 방식



- 주소형 변환 후 메모리접근




```
#include <stdio.h>
void main ( )
{
      char   c = 10;

      printf("%d₩n",  c) ;
      printf("%d₩n",  *&c );
      printf("%d₩n",  *(int*)&c );
}
```

2.1 포인터의 선언

- 목차
  - 포인터의 선언
  - char형 포인터
  - int형 포인터
  - char형 포인터 변수의 메모리접근
  - int형 포인터 변수의 메모리접근

27 / 123

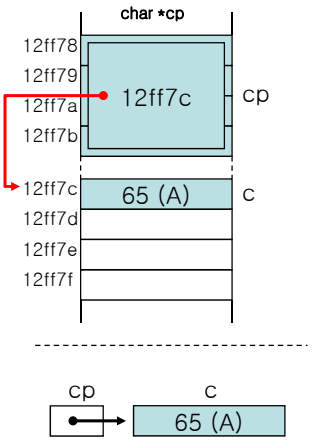
- 포인터의 선언

- 포인터는 주소를 저장하는 변수(메모리)
- *연산자를 사용
- char형 포인터 선언
  - 예) char c ;
  - 예) char *cp ;
- int형 포인터 선언
  - 예) int n ;
  - 예) int *np ;

28 / 123

- char형 포인터

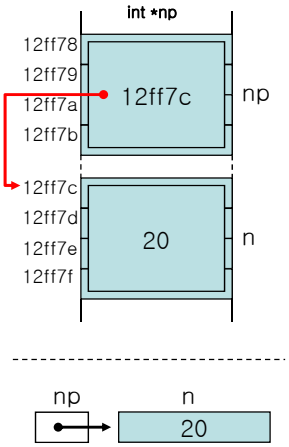
```
#include <stdio.h>
void main ( )
{
    char   c = 'A';
    char   *cp = &c;


    printf("%c %c₩n",  c,  *cp );
    printf("%d %d₩n",  sizeof(char),  sizeof(char*) );
                          //    1            4
    printf("%d %d₩n",  sizeof(c),  sizeof(cp) );
}
```



29 / 123

- int형 포인터

```
#include <stdio.h>
void main ( )
{
  int   n = 20;
  int   *np = &n;


  printf("%c %c₩n", n,  *np );
  printf("%d %d₩n",  sizeof(int),  sizeof(int*) );
                //          4           4
  printf("%d %d₩n",  sizeof(n),  sizeof(np) );
}
```

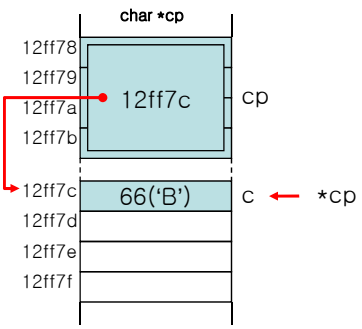


30 / 123

-char형 포인터 변수의 메모리접근

- * 연산자 사용

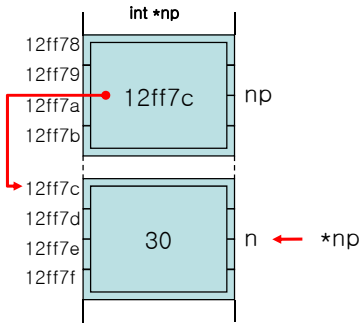
```
#include <stdio.h>
void main ( )
{
      char   c = 'A';
      char   *cp = &c;

      *cp = 'B';

      printf("%c %c₩n", c, *cp);
}
```



31 / 123

-int형 포인터 변수의 메모리 접근

```
#include <stdio.h>
void main ( )
{
      int    n = 20;
      int   *np = &n;

      *np = 30;
      printf("%d %d₩n", n, *np);
}
```



32 / 123

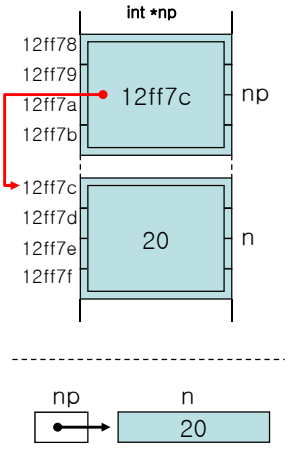
2.2 다차원 포인터

- 목차
  - 1차원 포인터
  - 2차원 포인터
  - 3차원 포인터

33 / 123

- 1차원 포인터

```
#include <stdio.h>
void main ( )
{
      int   n = 20;
      int   *np = &n;

      printf("%d %d₩n", n, *np);
}
```



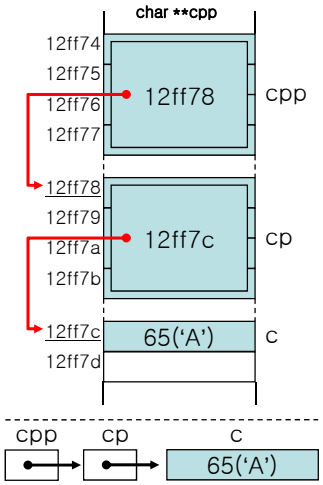
34 / 123

– 2차원 포인터

- char

```
#include <stdio.h>
void main ( )
{
      char   c = 'A';
      char   *cp;
      char   **cpp;

      cp = &c;
      cpp = &cp;

      printf("%c %c %c₩n", c, *cp, **cpp);

      printf("%x %x %x  %x ₩n",
            &c, cp, *cpp,  cpp  );

}
```

cpp하나 이전단계가
가지고 있는 주소값



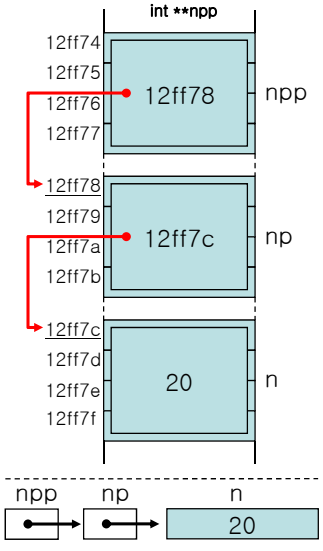
35 / 123

– 2차원 포인터

- int

```
#include <stdio.h>
void main ( )
{
  int   n = 20;
  int   *np;
  int   **npp;

  np = &n;
  npp = &np;

  printf("%d %d %d₩n", n, *np, **npp);
  printf("%x %x %x₩n", &n, np, *npp);
}
```

npp하나 이전단계가
가지고 있는 주소값



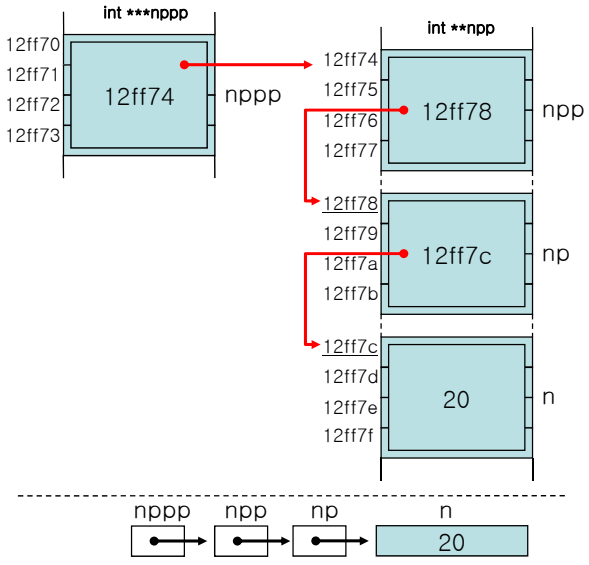
36 / 123

- 3차원 포인터

```
#include <stdio.h>
void main ( )
{
  int    n = 20;
  int    *np;
  int    **npp;
  int    ***nppp;

  np = &n;
  npp = &np;
  nppp = &npp;

  printf("%d %d %d %d₩n",  n,  *np,  **npp,  ***nppp);
  printf("%x %x %x %x₩n",  &n,  np,  *npp,  **nppp);
  printf("%x %x %x ₩n",  &np,  npp,  *npp);
  printf("%d %d %d %d₩n",  sizeof(int), sizeof(int*), sizeof(int**), sizeof(int***));
  printf("%d %d %d %d₩n",  sizeof(n), sizeof(np), sizeof(npp), sizeof(nppp));
}
```

37 / 123

- 3차원 포인터



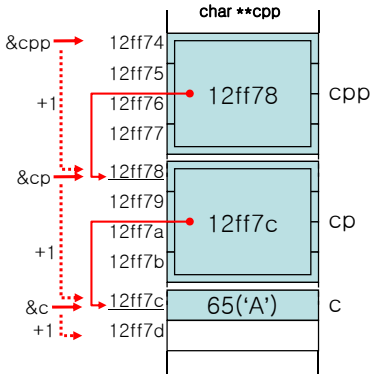
38 / 123

2.3 다차원 포인터의 가감산

- 목차
  - char형 주소의 연산
  - int형 주소의 연산
  - 다른 자료형 주소의 연산

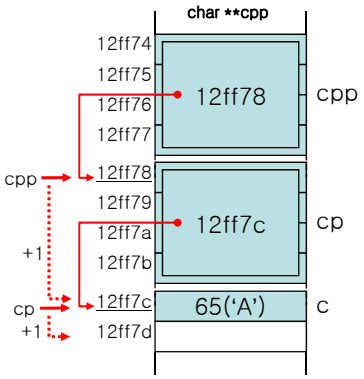
- char형 주소의 연산

```
#include <stdio.h>
void main ( )
{
      char   c = 'A';
      char   *cp;
      char   **cpp;

      cp = &c;
      cpp = &cp;

      printf("%x %x %x₩n",  &c,  &cp,  &cpp);
      printf("%x %x %x₩n",  &c+1,  &cp+1,  &cpp+1);
}
```



- char형 주소의 연산

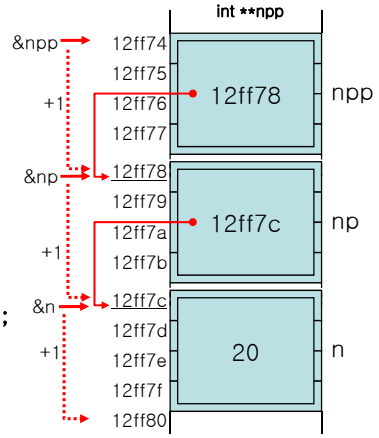
```
#include <stdio.h>
void main ( )
{
      char    c = 'A';
      char    *cp;
      char    **cpp;

      cp = &c;
      cpp = &cp;

      printf("%x %x %x₩n",  c,  cp,  cpp);
                              //  41
      printf("%x %x %x₩n",  c+1,  cp+1,  cpp+1);
                              //   42
}
```



41 / 123

- int형 주소의 연산

```
#include <stdio.h>
void main ( )
{
    int    n = 20;
    int    *np;
    int    **npp;

    np = &n;
    npp = &np;

    printf("%x %x %x₩n", &n, &np, &npp);
    printf("%x %x %x₩n", &n+1, &np+1, &npp+1);

}
```



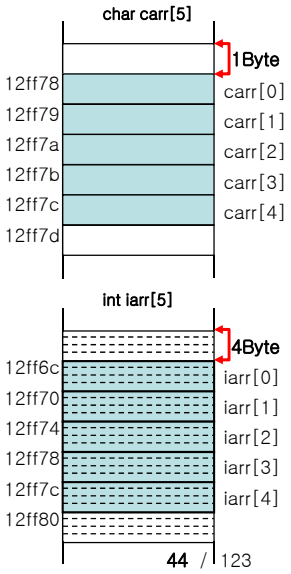
42 / 123

3.1 배열의 선언

- 목차
  - 배열의 선언
  - char형 배열
  - int형 배열
  - * 연산자와 [] 연산자, &연산자
  - 상수주소를 이용한 메모리 접근
  - 배열 이름과 배열 주소의 의미

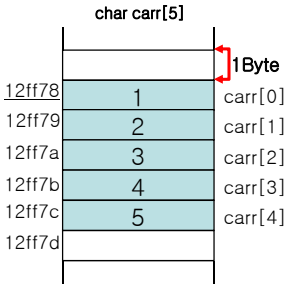
- 배열의 선언

- 배열은 자료형의 연속적인 메모리 공간
- [] 연산자를 사용
- char형 배열 선언
  - 예) char c ;
  - 예) char carr[5];
- int형 배열 선언
  - 예) int n ;
  - 예) int iarr[5];



- char형 배열

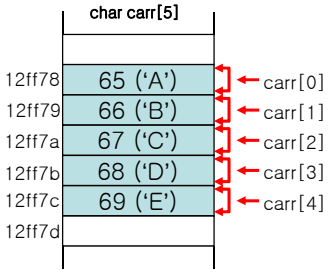
carr = &carr[0] =




```
#include <stdio.h>
void main ( )
{
      char   carr[5] = {1, 2, 3, 4, 5};

      printf( "%x %x₩n",  carr,  &carr[0] );
}
```

45 / 123

- char형 배열



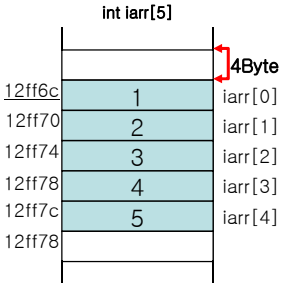

```
#include <stdio.h>
void main ( )
{
      char  carr[5] = {'A', 'B', 'C', 'D', 'E'};

      printf("%c %c %c %c %c₩n",
            carr[0], carr[1], carr[2], carr[3], carr[4]);
}
```

46 / 123

- int형 배열

```
#include <stdio.h>
void main ( )
{
      int  iarr[5] = { 1, 2, 3, 4, 5 };

      printf("%x %x₩n",  iarr,  &iarr[0] );
}
```

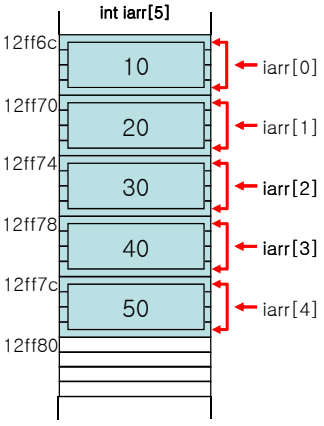
iarr = &iarr[0] =



47 / 123

- int형 배열

```
#include <stdio.h>
void main( )
{
      int   iarr[5] ={10, 20, 30, 40, 50};


      printf("%d %d %d %d %d₩n",
            iarr[0], iarr[1], iarr[2], iarr[3], iarr[4]);
}
```

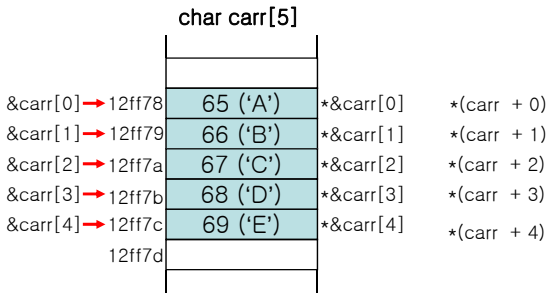


48 / 123

– *연산자와 [ ]연산자, &연산자

```
#include <stdio.h>
void main ( )
{
    char   carr[5] = {'A', 'B', 'C', 'D', 'E'};

    printf("%c %c %c %c %c₩n",   carr[0], carr[1], carr[2], carr[3], carr[4]);
    printf("%c %c %c %c %c₩n",
                *carr, *(carr+1), *(carr+2), *(carr+3), *(carr+4));
    printf("%c %c %c %c %c₩n",
                *&carr[0], *&carr[1], *&carr[2], *&carr[3], *&carr[4]);
}
```


– *연산자와 [ ]연산자, &연산자

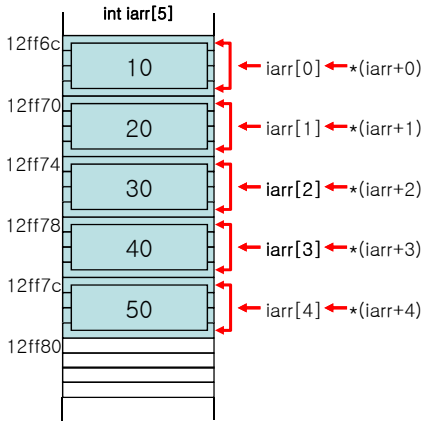


– *연산자와 [ ]연산자, &연산자

```
#include <stdio.h>
void main( )
{
      int   iarr[5] ={10, 20, 30, 40, 50};

      printf("%d %d %d %d %d₩n",   iarr[0], iarr[1], iarr[2], iarr[3], iarr[4]);

      printf("%d %d %d %d %d₩n",
                  *(iarr+0), *(iarr+1), *(iarr+2), *(iarr+3), *(iarr+4));

      printf("%d %d %d %d %d₩n",
                  *&iarr[0], *&iarr[1], *&iarr[2], *&iarr[3], *&iarr[4]);
}
```

51 / 123

– *연산자와 [ ]연산자, &연산자



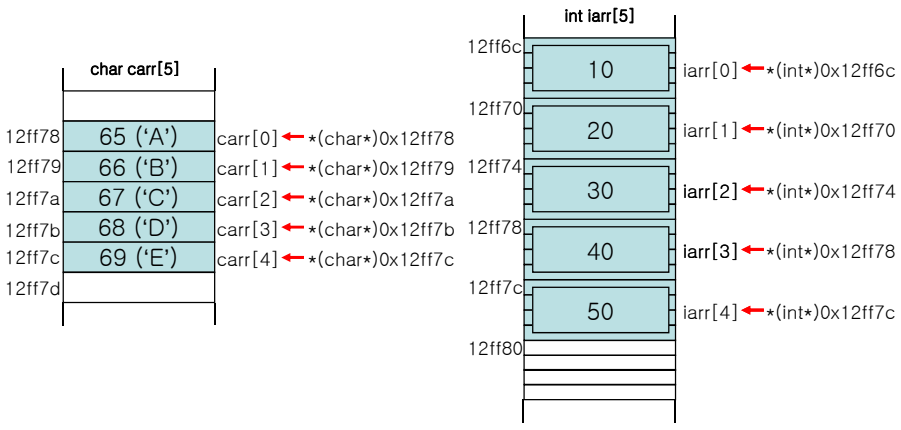
52 / 123

-상수주소를 이용한 메모리 접근

```
#include <stdio.h>
void main ( )
{
      char   carr[5] = {'A', 'B', 'C', 'D', 'E'};

      printf("%c %c %c %c %c₩n",
            *(char*)0x12ff78, *(char*)0x12ff79, *(char*)0x12ff7a,
            *(char*)0x12ff7b, *(char*)0x12ff7c);
}
```

```
#include <stdio.h>
void main( )
{
      int iarr[5] ={10, 20, 30, 40, 50};

      printf("%d %d %d %d %d₩n",
            *(int*)0x12ff6c, *(int*)0x12ff70, *(int*)0x12ff74,
            *(int*)0x12ff78, *(int*)0x12ff7c);
}
```


-상수주소를 이용한 메모리 접근



- 배열 이름과 배열주소의 의미

```
#include <stdio.h>
void main ( )
{
      char   carr[5] = {'A', 'B', 'C', 'D', 'E'};

      printf("%x %x %x %x %x₩n",    carr, carr+1, carr+2, carr+3, carr+4);
}
```

```
#include <stdio.h>
void main( )
{
      int   iarr[5] ={10, 20, 30, 40, 50};

      printf("%x %x %x %x %x₩n",     iarr, iarr+1, iarr+2, iarr+3, iarr+4);
}
```

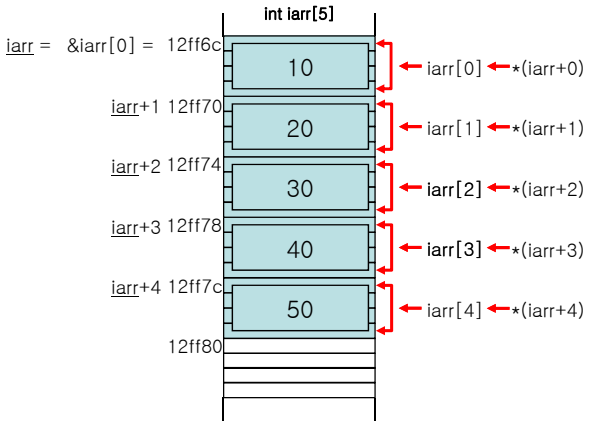
55 / 123

3.2 다차원 배열

- 목차
  - 1차원 배열 복습
  - 다차원 배열의 선언
  - char형 2차원 배열
  - int형 2차원 배열
  - 배열의 주소와 크기

56 / 123

- 1차원 배열 복습



57 / 123

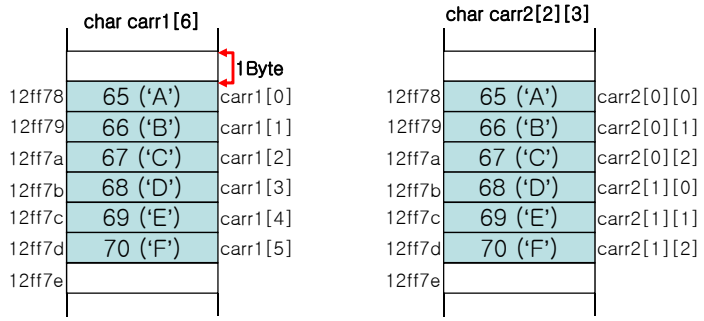
- 다차원 배열의 선언

- []... 연산자 사용
- char형 2차원 배열 선언
  - 예) char carr1[6] ;
  - 예) char carr2[2][3];

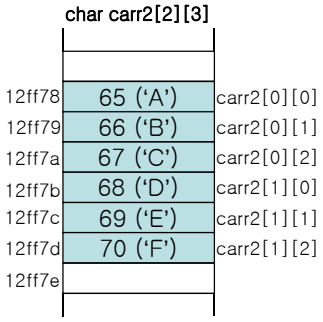
- int형 2차원 배열 선언
  - 예) int iarr1[9] ;
  - 예) int iarr2[3][3];

58 / 123

- 다차원 배열의 선언



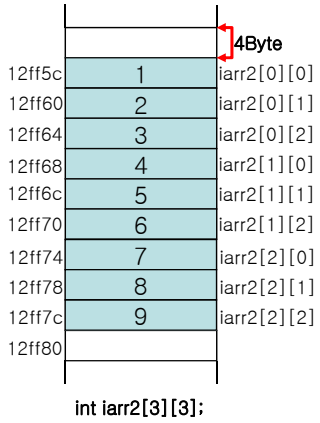
- char형 2차원 배열




```
#include <stdio.h>
void main ( )
{
      char carr2[2][3] = {'A','B','C','D','E','F'};

      printf("%c %c %c %c %c %c₩n",
            carr2[0][0], carr2[0][1], carr2[0][2],
            carr2[1][0], carr2[1][2], carr2[1][2]);
}
```

- int형 2차원 배열

```
#include <stdio.h>
void main( )
{
      int   iarr2[3][3] = { 1,2,3,4,5,6,7,8,9};


      printf("%x₩n", iarr2);
      printf("%d %d %d %d %d %d %d %d %d₩n",
            iarr2[0][0], iarr2[0][1], iarr2[0][2],
            iarr2[1][0], iarr2[1][1], iarr2[1][2],
            iarr2[2][0], iarr2[2][1], iarr2[2][2]);
}
```



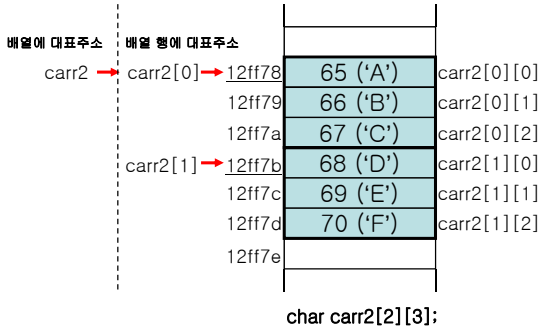
61 / 123

- 배열의 주소와 크기

```
#include <stdio.h>
void main( )
{
      char   carr2[2][3] = {'A','B','C','D','E','F'};


      printf("%x ₩n", carr2);

      printf("%x %x %x %x ₩n",
            carr2[0],  carr2[1],  &carr2[0][0],  &carr2[1][0]);

      printf("%d %d %d ₩n", sizeof(carr2),
            sizeof(carr2[0]),   sizeof(carr2[1]));   // 6  3  3

}
```

62 / 123

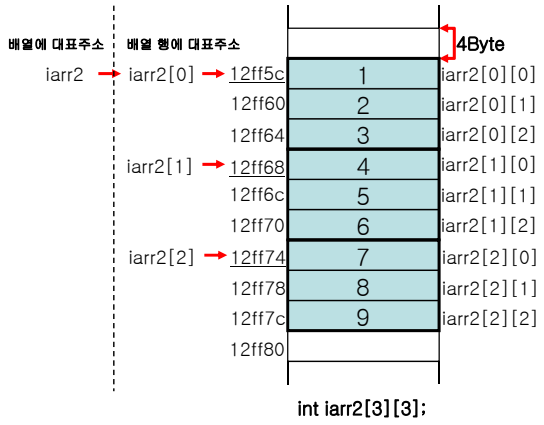
- 배열의 주소와 크기



- 배열의 주소와 크기

```
#include <stdio.h>
void main ( )
{
      int   iarr2[3][3] = {1,2,3,4,5,6,7,8,9};

      printf("%x₩n", iarr2);

      printf("%x %x %x₩n", iarr2[0], iarr2[1], iarr2[2]);

      printf("%d %d %d %d₩n", sizeof(iarr2), sizeof(iarr2[0]),
            sizeof(iarr2[1]), sizeof(iarr2[2])); //  36   12   12   12
}
```

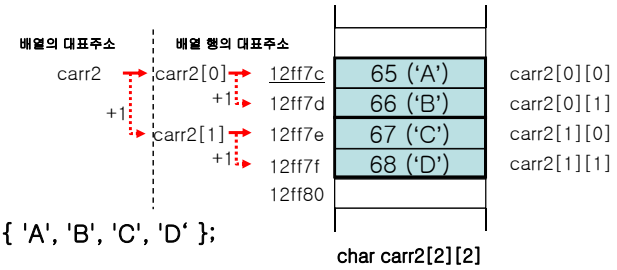
- 배열의 주소와 크기



3.3 다차원 주소의 의미

- 목차
  - char형 배열주소의 연산
  - char형 2차원 배열주소와 * 연산자
  - char형 3차원 배열주소와 * 연산자
  - int형 3차원 배열

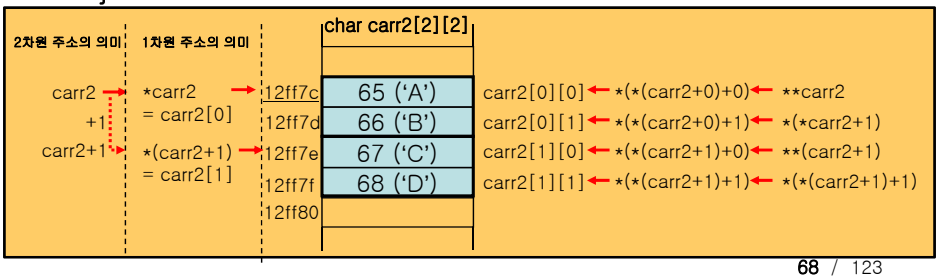
- char형 배열주소의 연산




```
#include <stdio.h>
void main ( )
{
      char   carr2[2][2] = { 'A', 'B', 'C', 'D' };

      printf("%x %x %x %x₩n",
            &carr2[0][0], carr2, carr2[0], carr2[1]);

      printf("%x %x %x %x₩n",
            &carr2[0][0]+1, carr2+1,  carr2[0]+1, carr2[1]+1);
}           //  12ff7d        12ff7c     12ff7d        12ff7f
```


- char형 2차원 배열주소와 * 연산자

```
#include <stdio.h>
void main ( )  {
      char   carr2[2][2] = { 'A', 'B', 'C', 'D' };
      printf("%x %x %x₩n", carr2+1, carr2[1], carr2[1][0]);
      printf("%c %c %c %c₩n",
            carr2[0][0], carr2[0][1], carr2[1][0], carr2[1][1]);

      printf("%x %x %x₩n", carr2+1, *(carr2+1), **(carr2+1));
      printf("%c %c %c %c₩n",
            **carr2, *(*carr2+1), **(carr2+1), *(*(carr2+1)+1));
}
```

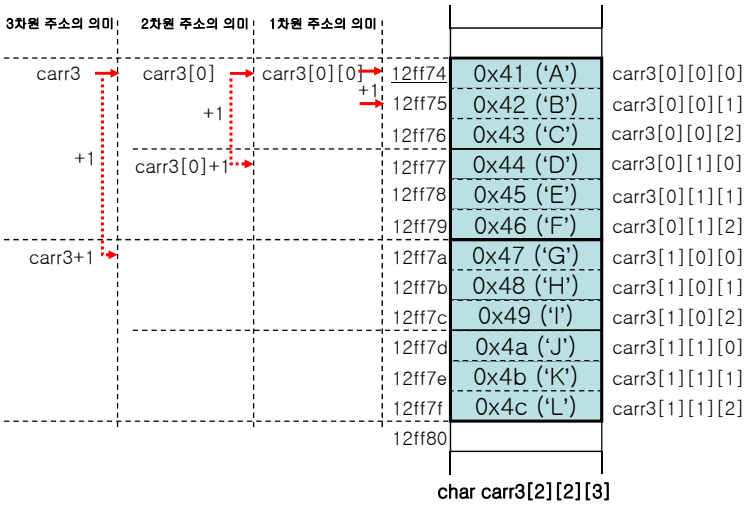


- char형 2차원 배열주소와 * 연산자

```
#include <stdio.h>
void main ( )
{
  char carr3[2][2][3] = {'A','B','C',   // [0][0][0], [0][0][1], [0][0][2],
                         'D','E','F',   // [0][1][0], [0][1][1], [0][1][2],
                         'G','H','I',   // [1][0][0], [1][0][1], [1][0][2],
                         'J','K','L' }; // [1][1][0], [1][1][1], [1][1][2],

        printf("%x %x %x %x₩n",
                carr3, carr3[0], carr3[0][0], carr3[0][0][0]);
        printf("%x %x %x %x₩n",
                carr3+1, carr3[0]+1, carr3[0][0]+1, carr3[0][0][0]+1);
        printf("%x %x %x %x₩n",
                carr3+1, *carr3+1, **carr3+1, ***carr3+1);
}
```

69 / 123

-char형 3차원 배열주소와 * 연산자

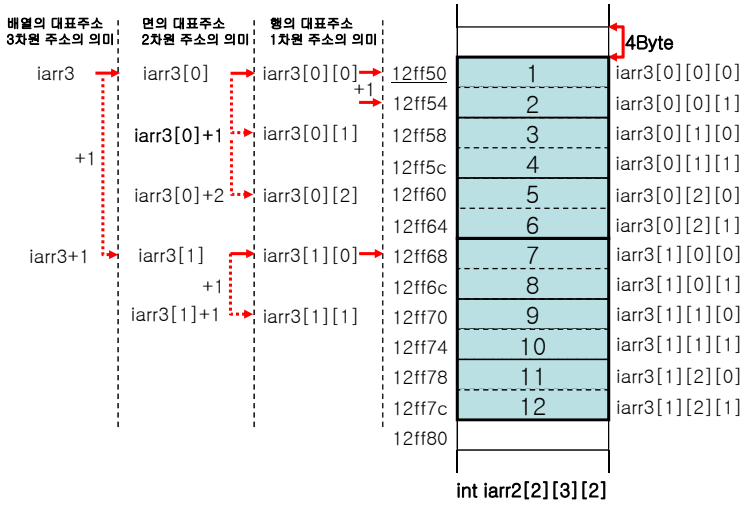


70 / 123

- int형 3차원 배열

```
#include <stdio.h>
void main ( )
{
      int iarr3[2][3][2] = {1,2,3,4,5,6,7,8,9,10,11,12};

      printf("%x₩n", iarr3); //
      printf("%x %x₩n",iarr3[0], iarr3[1]);
      printf("%x %x %x %x₩n",
            iarr3[0][0], iarr3[0][1], iarr3[1][0], iarr3[1][1]);
      printf("%x %x %x %x₩n",
            &iarr3[0][0][0], &iarr3[0][1][0], &iarr3[1][0][0],&iarr3[1][1][0]);
      printf("%x %x %x %x₩n",
            iarr3[0][0][0], iarr3[0][1][0], iarr3[1][0][0], iarr3[1][1][0]);
        //       1              3                  7             9
}
```

- int형 3차원 배열



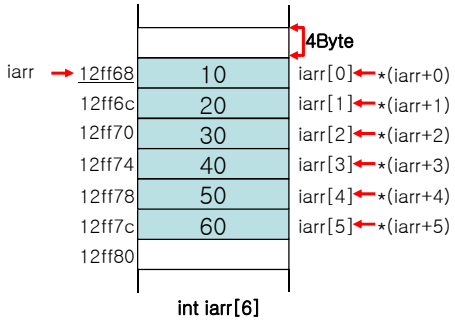
3.4 배열 요소의 접근

- 목차
  - int형 1차원 배열요소의 접근(*)
  - int형 1차원 배열요소의 접근([])
  - int형 2차원 배열요소의 접근(*)
  - int형 2차원 배열요소의 접근([])
  - 상수주소를 이용한 int형 2차원 배열

-int형 1차원 배열요소의 접근(*)

```
#include <stdio.h>

void main( )
{
      int    iarr[6] = { 10, 20, 30, 40, 50, 60 };

      printf("%x₩n", iarr);

      printf("%d %d %d %d %d %d₩n",
            iarr[0], iarr[1], iarr[2], iarr[3], iarr[4], iarr[5]);

      printf("%d %d %d %d %d %d₩n",
            *iarr, *(iarr+1), *(iarr+2), *(iarr+3), *(iarr+4), *(iarr+5));
}
```

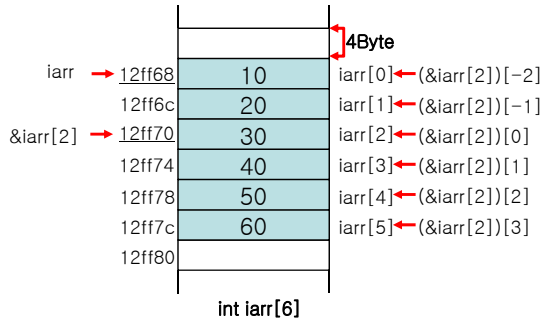
-int형 1차원 배열요소의 접근(*)



-int형 1차원 배열요소의 접근([])

```
#include <stdio.h>

void main( )
{
      int   iarr[6] = {10, 20, 30, 40, 50, 60 };

      printf("%x₩n", &iarr[2]);// 12ff70
      printf("%d %d %d %d %d %d₩n",
            iarr[0], iarr[1], iarr[2], iarr[3], iarr[4], iarr[5]);
      printf("%d %d %d %d %d %d₩n", (&iarr[2])[-2], (&iarr[2])[-1],
            (&iarr[2])[0], (&iarr[2])[1], (&iarr[2])[2], (&iarr[2])[3]);
}
```

-int형 1차원 배열요소의 접근([])



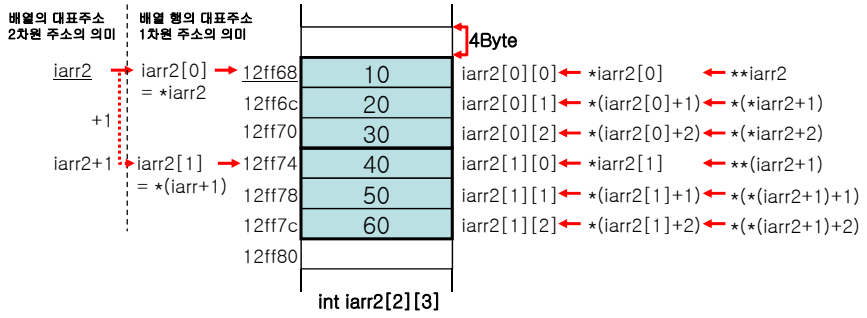
77 / 123

-int형 2차원 배열요소의 접근(*)

```
#include <stdio.h>

void main( )
{
	int iarr2[2][3] = { 10, 20, 30, 40, 50, 60 };

	printf("%x₩n",  iarr2 );
	printf("%d %d %d %d %d %d₩n", iarr2[0][0], iarr2[0][1], iarr2[0][2],
		iarr2[1][0], iarr2[1][1], iarr2[1][2] );

	printf("%d %d %d %d %d %d₩n", *iarr2[0], *(iarr2[0]+1), *(iarr2[0]+2),
		*iarr2[1], *(iarr2[1]+1), *(iarr2[1]+2) );

	printf("%d %d %d %d %d %d₩n", **iarr2, *(*iarr2+1), *(*iarr2+2),
		**(iarr2+1), *(*(iarr2+1)+1), *(*(iarr2+1)+2) );
}
```

78 / 123

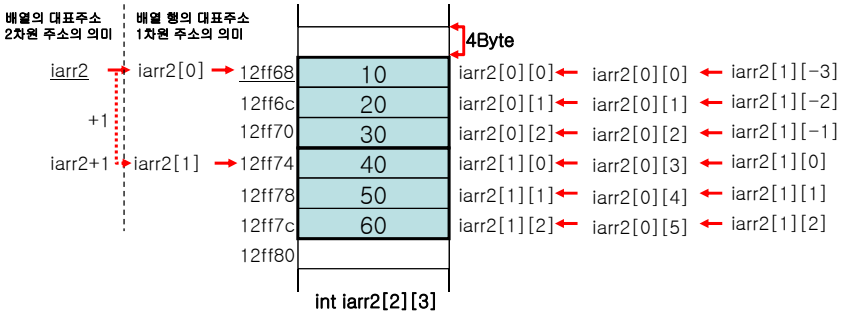
-int형 2차원 배열요소의 접근(*)



-int형 2차원 배열요소의 접근([])

```
#include <stdio.h>
void main( )
{
	int	iarr2[2][3] = { 10, 20, 30, 40, 50, 60 };

	printf("%x₩n", iarr2);
	printf("%d %d %d %d %d %d₩n", iarr2[0][0], iarr2[0][1], iarr2[0][2],
		iarr2[1][0], iarr2[1][1], iarr2[1][2]);

	printf("%d %d %d %d %d %d₩n", iarr2[0][0], iarr2[0][1], iarr2[0][2],
		iarr2[0][3], iarr2[0][4], iarr2[0][5]);

	printf("%d %d %d %d %d %d₩n", iarr2[1][-3], iarr2[1][-2], iarr2[1][-1],
		iarr2[1][0], iarr2[1][1], iarr2[1][2]);
}
```

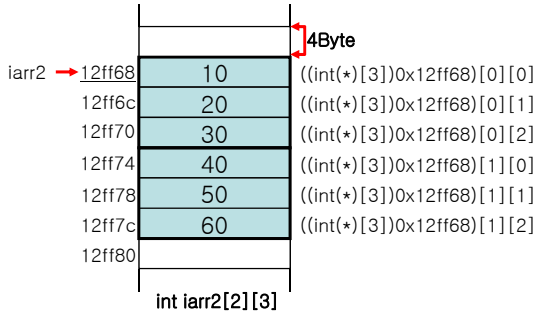
-int형 2차원 배열요소의 접근([])



-상수주소를 이용한 int형 2차원 배열

```
#include <stdio.h>
void main( )
{
      int   iarr2[2][3] = { 10, 20, 30, 40, 50, 60 };

      printf("%x₩n", iarr2);
      printf("%d %d %d %d %d %d₩n", iarr2[0][0], iarr2[0][1], iarr2[0][2],
            iarr2[1][0], iarr2[1][1], iarr2[1][2]);

      printf("%d %d %d %d %d %d₩n",
            ((int(*)[3])0x12ff68)[0][0],((int(*)[3])0x12ff68)[0][1],
            ((int(*)[3])0x12ff68)[0][2], ((int(*)[3])0x12ff68)[1][0],
            ((int(*)[3])0x12ff68)[1][1],((int(*)[3])0x12ff68)[1][2]);
}
```

-상수주소를 이용한 int형 2차원 배열

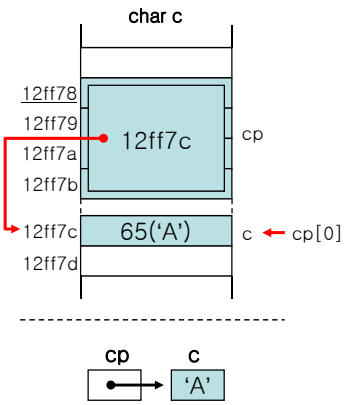


4.1 1차원 포인터와 배열

- 목차
  - char형 변수의 주소를 포인터에 저장
  - char형 배열의 주소를 포인터에 저장
  - 배열과 포인터의 차이점
  - int형 배열의 주소를 포인터에 저장
  - 포인터를 이용한 배열요소 접근

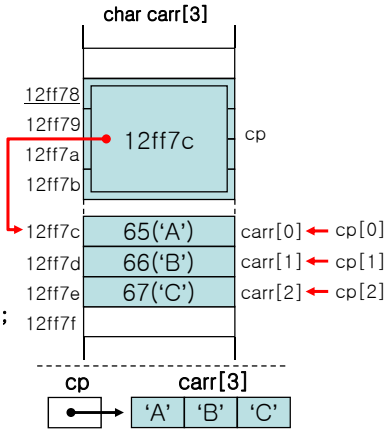
-char형 변수의 주소를 포인터에 저장

```
#include <stdio.h>
void main ( )
{
      char   c = 'A';
      char   *cp = &c;

      printf("%c %c %c₩n", c, *cp, cp[0]);
}
```



-char형 배열의 주소를 포인터에 저장

```
#include <stdio.h>
void main ( )
{
      char carr[3] ={'A', 'B', 'C'};
      char *cp = carr;

      printf("%c %c %c₩n", carr[0], carr[1], carr[2]);
      printf("%c %c %c₩n", *carr, *(carr+1), *(carr+2));

      printf("%c %c %c₩n", cp[0], cp[1], cp[2]);
      printf("%c %c %c₩n", *cp, *(cp+1), *(cp+2));
}
```



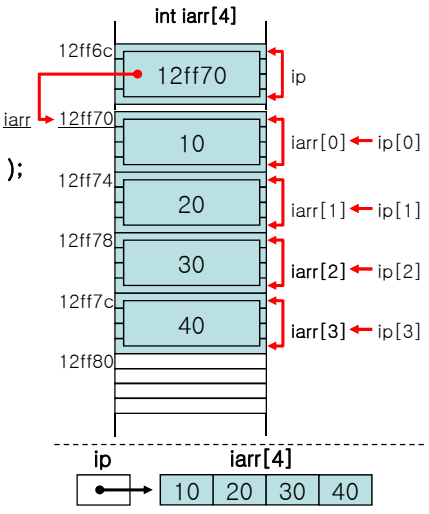
- 배열과 포인터의 차이점

```
#include <stdio.h>
void main ( )
{
      char carr[3] ={'A', 'B', 'C'};
      char *cp = carr;

      printf("%d %d₩n", sizeof(cp), sizeof(carr) );

      cp = NULL;

      //carr = NULL; // 배열의 시작주소
}                          // 상수로 대입 불가.
```

87 / 123

-int형 배열의 주소를 포인터에 저장

```
#include <stdio.h>
void main ( )
{
      int   iarr[4] ={10,20,30,40};
      int   *ip = iarr;

      printf("%d %d₩n", sizeof(iarr), sizeof(ip) );

      printf("%d %d %d %d₩n",
          iarr[0], iarr[1], iarr[2], iarr[3]);

      printf("%d %d %d %d₩n",
          ip[0], ip[1], ip[2], ip[3]);
}
```

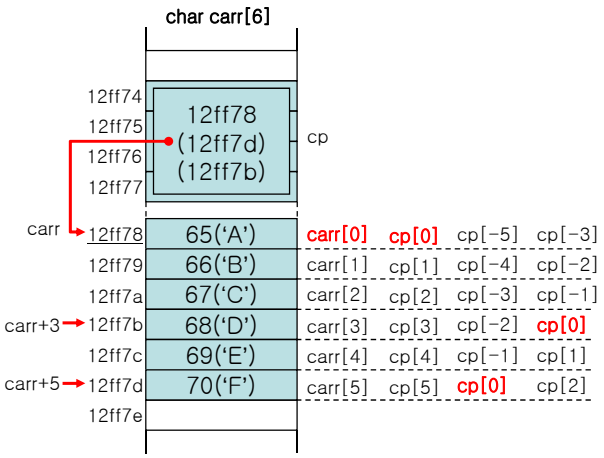


88 / 123

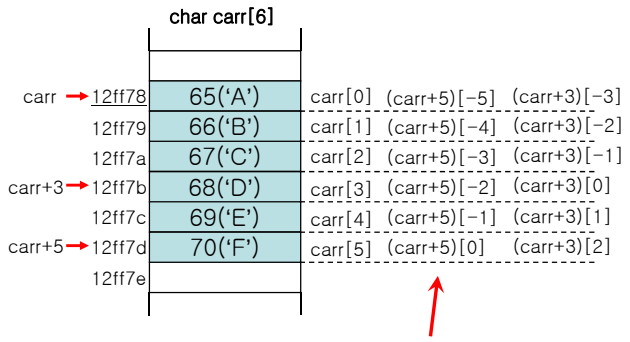
-포인터를 이용한 배열요소 접근

```
#include <stdio.h>
void main ( )
{
        char carr[6] = {'A','B','C','D','E','F'};
        char *cp = carr;

        printf("%c %c %c %c %c %c₩n",
                cp[0], cp[1], cp[2], cp[3], cp[4], cp[5]);

        cp = carr+5;
        printf("%c %c %c %c %c %c₩n",
                cp[0], cp[-1], cp[-2], cp[-3], cp[-4], cp[-5]);

        cp = carr+3;
        printf("%c %c %c %c %c %c₩n",
                cp[-3], cp[-2], cp[-1], cp[0], cp[1], cp[2]);
}
```


-포인터를 이용한 배열요소 접근



-포인터를 이용한 배열요소 접근




```
carr      ➔ 12ff78 ,  1차원 배열의 주소
carr + 5  ➔ 12ff7d ,
            (carr + 5)[-5] ➔ *((carr + 5) - 5)  ➔ carr[0]
```

91 / 123

4.2 1차원 배열을 다차원 배열처럼 사용하기

- 목차
  - char형 1차원 배열을 2차원 배열처럼 접근하는 포인터
  - int형 1차원 배열을 2차원 배열처럼 접근하는 포인터
  - char형 2차원 배열을 2차원 배열처럼 접근하는 포인터
  - 다차원 배열처럼 접근하는 포인터

92 / 123

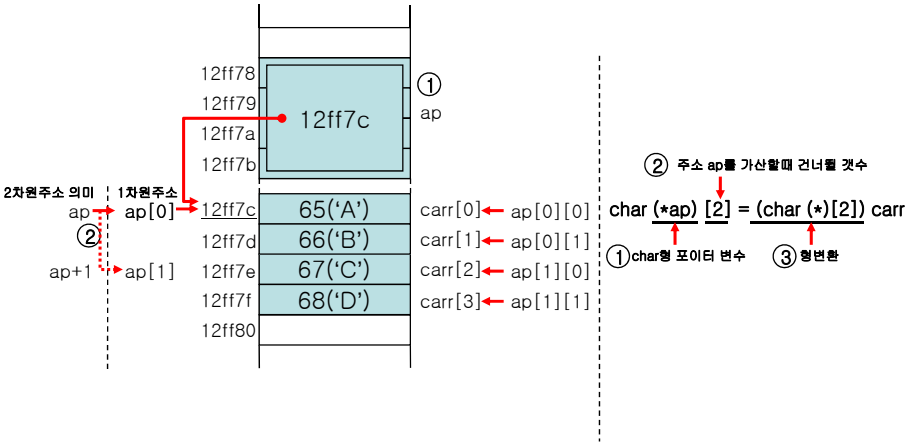
-char형 1차원 배열을 2차원 배열처럼 접근하는 포인터

```
#include <stdio.h>
void main ( )
{
        char carr[4]={'A','B','C','D'};
        char (*ap)[2] = (char(*)[2]) carr;
```

열의 갯수

```
        printf("%c %c %c %c₩n", carr[0],  carr[1],  carr[2],  carr[3]);
        printf("%c %c %c %c₩n", ap[0][0],  ap[0][1],  ap[1][0], ap[1][1]);
}
```

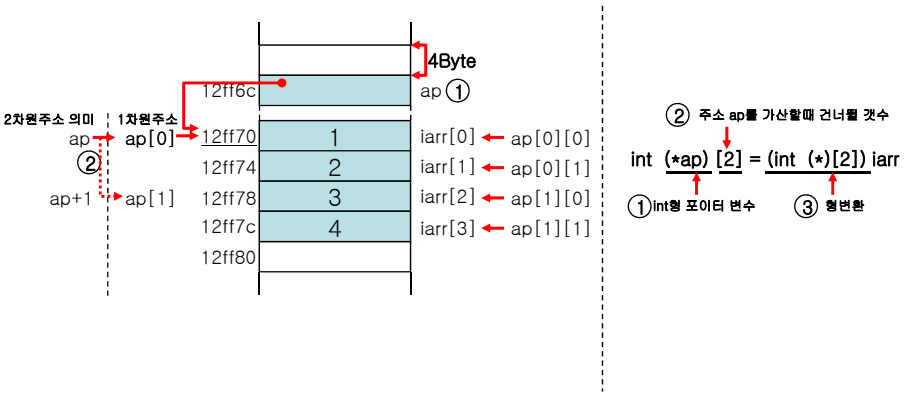
-char형 1차원 배열을 2차원 배열처럼 접근하는 포인터



- int형 1차원 배열을 2차원 배열처럼 접근하는 포인터

```
#include <stdio.h>
void main ( )
{
        int iarr[4]={1,2,3,4};
        int (*ap)[2] = (int(*)[2]) iarr;


        printf("%d %d %d %d₩n",iarr[0], iarr[1], iarr[2], iarr[3]);
        printf("%d %d %d %d₩n",ap[0][0], ap[0][1], ap[1][0], ap[1][1]);
}
```

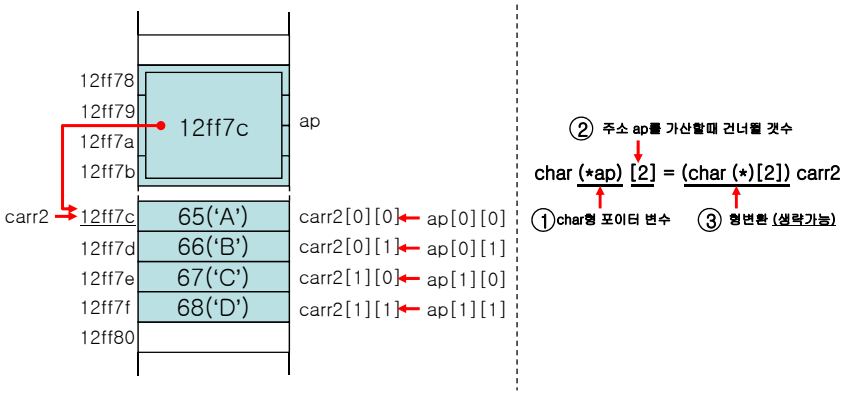

- int형 1차원 배열을 2차원 배열처럼 접근하는 포인터



-char형 2차원 배열을 2차원 배열처럼 접근하는 포인터

```
#include <stdio.h>
void main ( )
{
      char   carr2[2][2]={'A','B','C','D'};
      char   (*ap)[2] = (char(*)[2]) carr2;

      printf("%c %c %c %c₩n",carr2[0][0], carr2[0][1],
            carr2[1][0], carr2[1][1]);

      printf("%c %c %c %c₩n",ap[0][0], ap[0][1], ap[1][0], ap[1][1]);
}
```

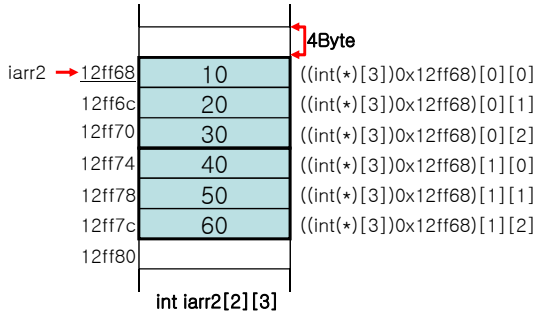

-char형 2차원 배열을 2차원 배열처럼 접근하는 포인터



-상수주소를 이용한 int형 2차원 배열

```
#include <stdio.h>
void main( )
{
      int   iarr2[2][3] = {10,20,30,40,50,60};

      printf("%x₩n", iarr2);

      printf("%d %d %d %d %d %d₩n", iarr2[0][0], iarr2[0][1], iarr2[0][2],
            iarr2[1][0], iarr2[1][1], iarr2[1][2]);

      printf("%d %d %d %d %d %d₩n",
            ((int(*)[3])0x12ff68)[0][0],((int(*)[3])0x12ff68)[0][1],
            ((int(*)[3])0x12ff68)[0][2], ((int(*)[3])0x12ff68)[1][0],
            ((int(*)[3])0x12ff68)[1][1],((int(*)[3])0x12ff68)[1][2]);
}
```

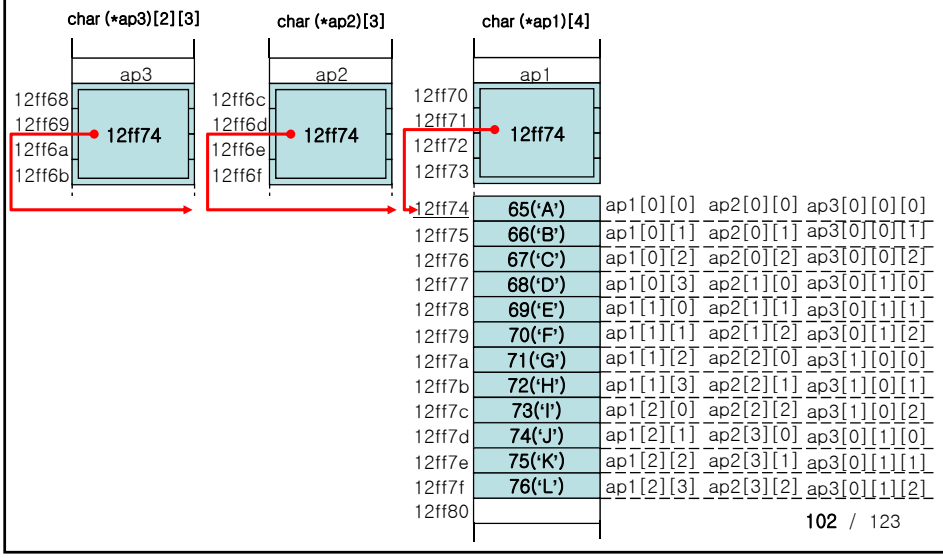

-상수주소를 이용한 int형 2차원 배열



- 다차원 배열처럼 접근하는 포인터

```
#include <stdio.h>
void main( )
{
      char carr[12]={'A','B','C','D','E','F','G','H','I','J','K','L'};

      char (*ap1)[4] = (char (*)[4])carr;
      char (*ap2)[3] = (char (*)[3])carr;
      char (*ap3)[2][3] = (char (*)[2][3])carr;

     printf("%c%c%c%c%c%c%c%c%c%c%c%c₩n",
            ap1[0][0],ap1[0][1],ap1[0][2],ap1[0][3],
            ap1[1][0],ap1[1][1],ap1[1][2],ap1[1][3],
            ap1[2][0],ap1[2][1],ap1[2][2],ap1[2][3]);

                              ...
}
```


- 다차원 배열처럼 접근하는 포인터



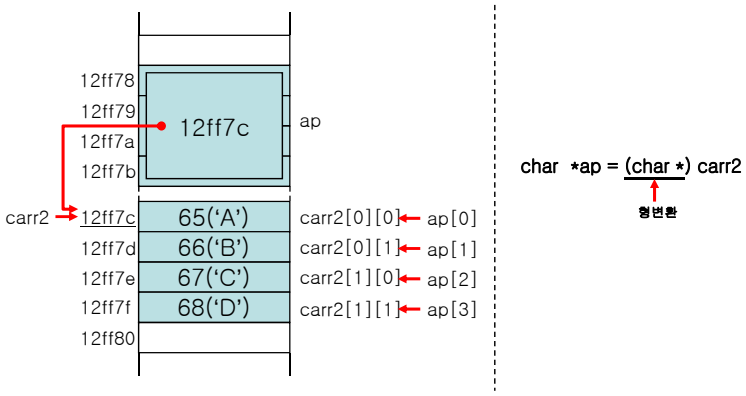
4.3 다차원 배열을 1차원 배열처럼 사용하기

- 목차
  - char형 2차원 배열을 1차원처럼 사용
  - int형 2차원 배열을 1차원처럼 사용
  - 배열과 포인터의 크기

- char형 2차원 배열을 1차원처럼 사용

```
#include <stdio.h>
void main ( )
{
      char   carr2[2][2] = { 'A', 'B', 'C', 'D' };
      char   *ap = carr2;

      printf("%c %c %c %c₩n",
            carr2[0][0],carr2[0][1],carr2[1][0],carr2[1][1]);

      printf("%c %c %c %c₩n",
            ap[0], ap[1], ap[2], ap[3]);
}
```

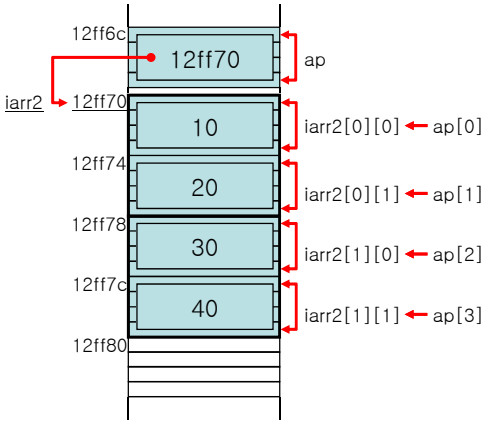
- char형 2차원 배열을 1차원처럼 사용



- char형 2차원 배열을 1차원처럼 사용

```
#include <stdio.h>
void main( )
{
      int   iarr2[2][2]={10,20,30,40};
      int   *ap = (int *)iarr2;

      printf("%d %d %d %d₩n",
            iarr2[0][0],iarr2[0][1],iarr2[1][0],iarr2[1][1]);

      printf("%d %d %d %d₩n",
            ap[0], ap[1], ap[2], ap[3]);
}
```

- char형 2차원 배열을 1차원처럼 사용



- 배열과 포인터의 크기

```
#include <stdio.h>
void main( )
{
      int   iarr2[2][2]={10,20,30,40};
      int   *ap = (int *)iarr2;

      printf("%d %d₩n", sizeof(iarr2), sizeof( ap ));
}
```

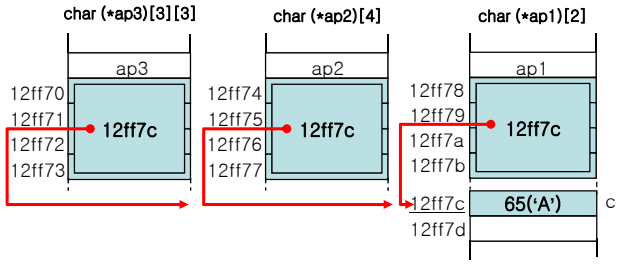
4.4 포인터와 배열의 이모저모

- 목차
  - char형 포인터변수의 연산
  - int형 포인터변수의 연산
  - char형 주소를 저장하는 배열
  - int형 주소를 저장하는 배열
  - 주소를 저장하는 배열의 크기
  - 1분 정리

- char형 포인터변수의 연산

```
#include <stdio.h>
void main ( )
{     char   c = 'A';
      char   (*ap1)[2] = (char (*)[2])&c;
      char   (*ap2)[4] = (char (*)[4])&c;
      char   (*ap3)[3][3] = (char (*)[3][3])&c;

      printf("%c %c %c₩n", **ap1, **ap2, ***ap3);  //   A   A   A
      printf("%c %c %c₩n", ap1[0][0], ap2[0][0], ap3[0][0][0]); //   A   A   A

    printf("%x %x %x %x₩n", ap1, ap1+1, ap1[0], ap1[1]);  // 주소
    printf("%x %x %x %x₩n", ap2, ap2+1, ap2[0], ap2[1]);  // 주소
    printf("%x %x %x %x₩n", ap3, ap3+1, ap3[0], ap3[1]);  // 주소

    printf("%x %x %x %x₩n", ap3[0], ap3[0]+1, ap3[0][0], ap3[0][1]);  // 주소
}
```

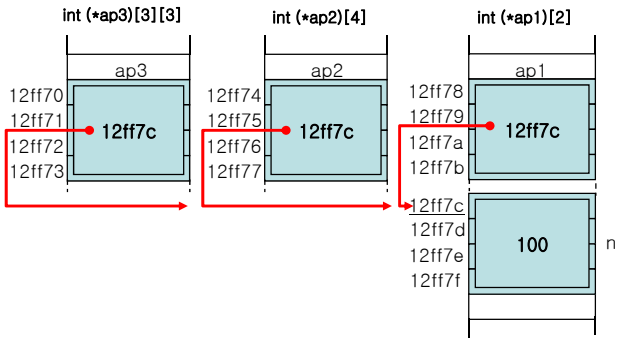
- char형 포인터변수의 연산



- char형 포인터변수의 연산

```
#include <stdio.h>
void main ( )
{
      int    n = 100;

      int   (*ap1)[2] = (int (*)[2])&n;
      int   (*ap2)[4] = (int (*)[4])&n;
      int   (*ap3)[3][3] = (int (*)[3][3])&n;

      printf("%d %d %d₩n", **ap1, **ap2, ***ap3);  // 100 100 100
      printf("%d %d %d₩n", ap1[0][0], ap2[0][0], ap3[0][0][0]);
                                    // 100 100 100

      printf("%x %x %x %x₩n", ap1, ap1+1, ap1[0], ap1[1]);  // 주소
      printf("%x %x %x %x₩n", ap2, ap2+1, ap2[0], ap2[1]); // 주소
      printf("%x %x %x %x₩n", ap3, ap3+1, ap3[0], ap3[1]); // 주소

      printf("%x %x %x %x₩n", ap3[0], ap3[0]+1, ap3[0][0], ap3[0][1]);
                              // 주소
}
```

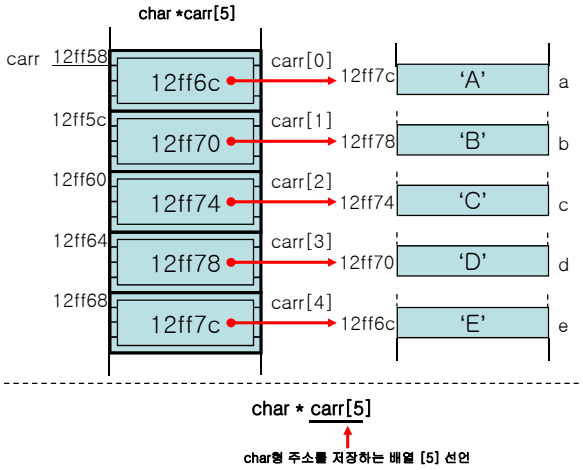
- char형 포인터변수의 연산



- char형 주소를 저장하는 배열

```
#include <stdio.h>
void main ( )
{
      char   a = 'A',  b = 'B',  c = 'C',  d = 'D',  e = 'E';
      char   *carr[5] ={&a, &b, &c, &d, &e};
        // char * 형 주소를 저장하는 배열 [5] 선언

      printf("%x %x %x %x %x₩n", &a, &b, &c, &d, &e);
      printf("%x %x %x %x %x₩n",
            carr[0], carr[1], carr[2], carr[3], carr[4]);

      printf("%c %c %c %c %c₩n", a, b, c, d, e);
      printf("%c %c %c %c %c₩n",
            *carr[0], *carr[1], *carr[2], *carr[3], *carr[4]);
}
```

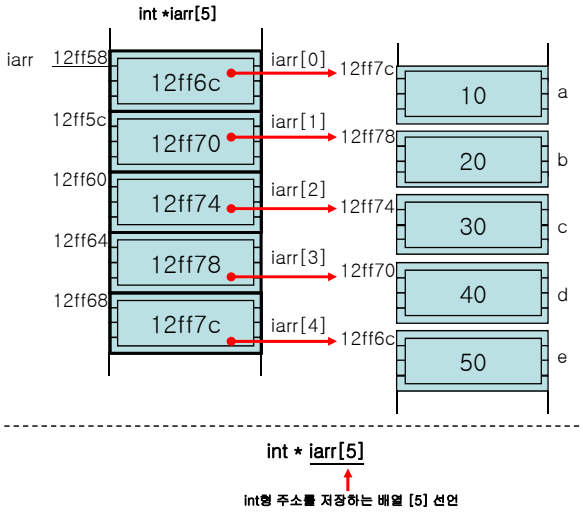
- char형 주소를 저장하는 배열



- int형 주소를 저장하는 배열

```
#include <stdio.h>
void main ( )
{
      int    a = 10, b = 20, c = 30, d = 40, e = 50;
      int    *iarr[5] ={&a, &b, &c, &d, &e};

      printf("%x %x %x %x %x₩n", &a, &b, &c, &d, &e);
      printf("%x %x %x %x %x₩n",
            iarr[0], iarr[1], iarr[2], iarr[3], iarr[4]);

      printf("%d %d %d %d %d₩n", a, b, c, d, e);
      printf("%d %d %d %d %d₩n",
            *iarr[0], *iarr[1], *iarr[2], *iarr[3], *iarr[4]);
}
```

- int형 주소를 저장하는 배열



- 주소를 저장하는 배열의 크기

```
#include <stdio.h>
void main ( )
{
      int   a = 10, b = 20, c = 30, d = 40, e = 50;
      int   *iarr[5] ={&a, &b, &c, &d, &e};

      printf("%d %d %d₩n", sizeof(iarr) , sizeof(iarr[0]), sizeof(*iarr[0]));
}
```

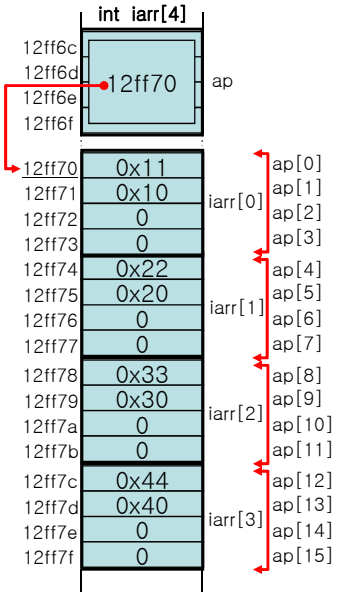
4.5 다른 형의 포인터 사용하기

- 목차
  - int형 배열의 주소를 저장하는 char형 포인터
  - char형 배열의 주소를 저장하는 int형 포인터

-int형 배열의 주소를 저장하는 char형 포인터

```
#include <stdio.h>
void main ( )
{
    int   iarr[4] = {0x1011, 0x2022, 0x3033, 0x4044};
    char   *ap = (char*) iarr;
      // int 형 배열의 주소를 저장하는 char 형 포인터

    printf("%x %x %x %x₩n", iarr[0], iarr[1], iarr[2], iarr[3]);
                              // 1011    2022    3033    4044
    printf("%x %x %x %x₩n", ap[0], ap[1], ap[2], ap[3]);    // 11  10  0  0
    printf("%x %x %x %x₩n", ap[4], ap[5], ap[6], ap[7]);    // 22  20  0  0
    printf("%x %x %x %x₩n", ap[8], ap[9], ap[10], ap[11]);  // 33  30  0  0
    printf("%x %x %x %x₩n", ap[12], ap[13], ap[14], ap[15]); // 44  40  0  0
}
```

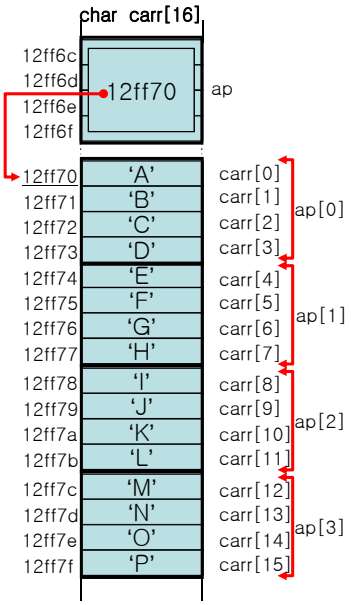
-int형 배열의 주소를 저장하는 char형 포인터



- char형 배열의 주소를 저장하는 int형 포인터

```
#include <stdio.h>
void main ( )
{
      char   carr[16] = {'A','B','C','D','E','F','G','H',
                         'I','J','K','L','M','N','O','P'};
      int   *ap = (int*) carr;
        // char 형 배열의 주소를 저장하는 int 형 포인터

      printf("%c %c %c %c₩n", carr[0],carr[1],carr[2],carr[3]); // A B C D
      printf("%c %c %c %c₩n", carr[4],carr[5],carr[6],carr[7]);
      printf("%c %c %c %c₩n", carr[8],carr[9],carr[10],carr[11]);
      printf("%c %c %c %c₩n", carr[12],carr[13],carr[14],carr[15]);

      printf("%x %x %x %x₩n", ap[0], ap[1], ap[2], ap[3]);
                  // 444344241   48474645  4C4B4A49  504F4E4D
}
```

-char형 배열의 주소를 저장하는 int형 포인터



123 / 123