10장 배열과 포인터

한빛미디어 Hanbit Media, Inc.

❖ 배열명으로 배열 요소 사용하기 (1/2)

배열명에 정수 연산을 수행하여 배열 요소 사용 소스 코드 예제10-1.c

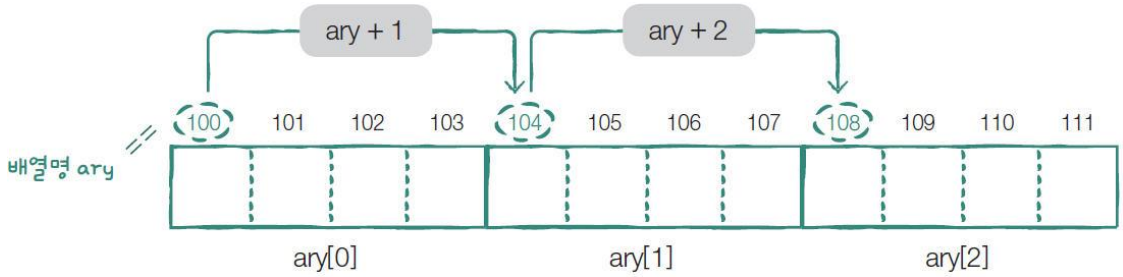
```c
#include <stdio.h>

int main(void)
{
    int ary[3];
    int i;

    *(ary + 0) = 10;
    *(ary + 1) = *(ary + 0) + 10;

    printf("세 번째 배열 요소에 키보드 입력 : ");
    scanf("%d", ary + 2);

    for (i = 0; i < 3; i++)
    {
        printf("%5d", *(ary + i));
    }

    return 0;
}
```

실행결과

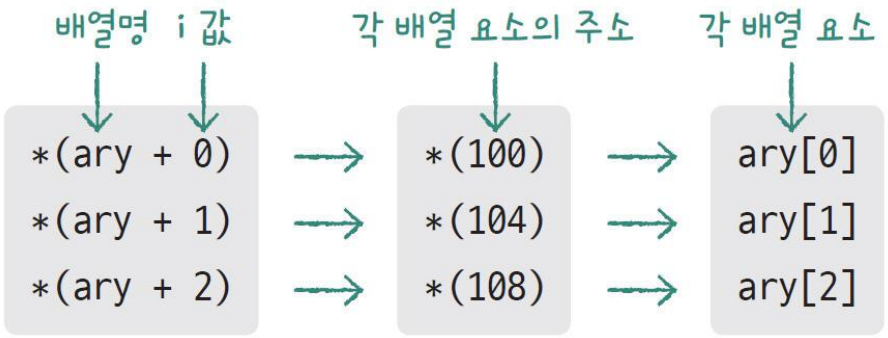
```
세 번째 배열 요소에 키보드 입력 : 30
   10   20   30
```

❖ 배열명으로 배열 요소 사용하기 (2/2)

- 배열명은 배열의 첫 번째 요소의 주소이다.



- 배열명에 정수를 더하고 *연산으로 모든 배열 요소를 사용한다.

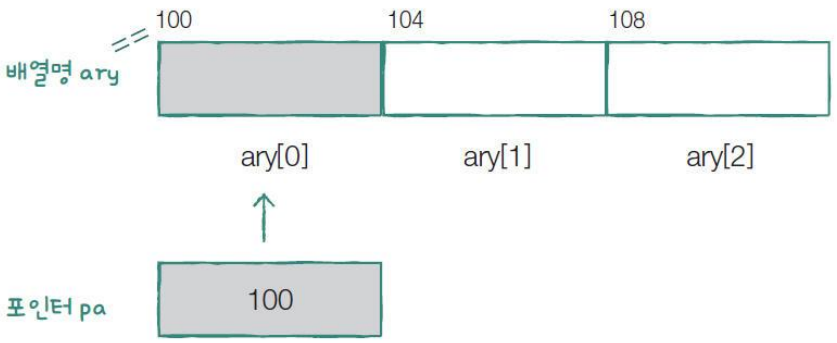


10- 1 배열과 포인터의 관계

❖ 배열명 역할을 하는 포인터 (1/2)

배열명처럼 사용되는 포인터 소스 코드 예제10-2.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int ary[3];
06      int *pa = ary;
07      int i;
08
09      *pa = 10;                  // 첫 번째 배열 요소에 10 대입
10      *(pa + 1) = 20;            // 두 번째 배열 요소에 20 대입
11      pa[2] = pa[0] + pa[1];     // 대괄호를 써서 pa를 배열명처럼 사용
12
```



❖ 배열명 역할을 하는 포인터 (2/2)

배열명처럼 사용되는 포인터 소스 코드 예제10-2.c

```
13      for (i = 0; i < 3; i++)
14      {
15          printf("%5d", pa[i]);
16      }
17
18      return 0;
19  }
```

실행결과

```
   10   20   30
```

```
pa[2] = pa[0] + pa[1]    // 11행
```

포인터 연산식

```
*(pa + 2) = *(pa + 0) + *(pa + 1)
```

pa에 저장된 값은 ary

```
*(ary + 2) = *(ary + 0) + *(ary + 1)
```

배열 요소 표현식

```
ary[2] = ary[0] + ary[1]
```

10- 1 배열과 포인터의 관계

❖ 배열명과 포인터의 차이 (1/2)

- sizeof 연산의 결과가 다르다.

```
int ary[3];
int *pa = ary;
sizeof(ary)     ←—— 12바이트, 배열 전체 크기
sizeof(pa)      ←—— 4바이트, 포인터 하나의 크기
```

- 상수와 변수의 차이가 있다.

배열명은 값을 바꿀 수 없음

```
ary = ary + 1
ary++
```

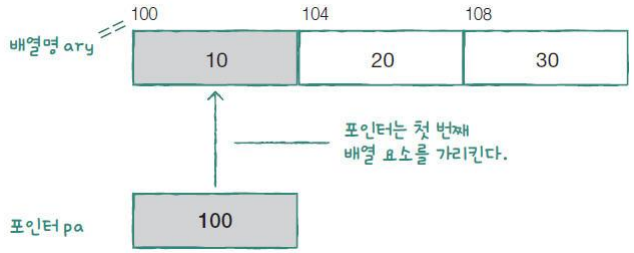
포인터는 값을 바꿀 수 있음

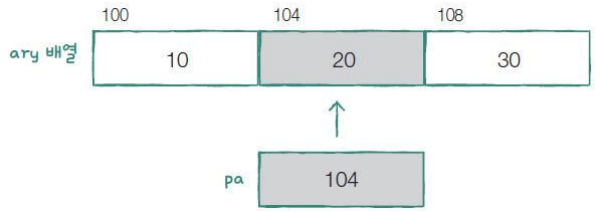
```
pa = pa + 1
pa++
```

❖ 배열명과 포인터의 차이 (2/2)

포인터를 이용한 배열의 값 출력 소스 코드 예제10-3.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int ary[3] = { 10, 20, 30 };
06      int *pa = ary;
07      int i;
08
09      printf("배열의 값 : ");
10      for (i = 0; i < 3; i++)
11      {
12          printf("%d ", *pa);    // pa가 가리키는 배열 요소 출력
13          pa++;                 // 다음 배열 요소를 가리키도록 pa 값 증가
14      }
15
16      return 0;
17  }
```





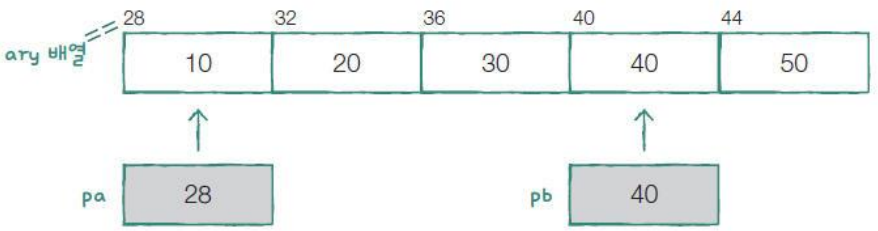
실행결과

```
배열의 값 : 10 20 30
```

❖ 포인터의 뺄셈과 관계 연산 (1/2)

포인터의 뺄셈과 관계 연산 소스 코드 예제10-4.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int ary[5] = { 10, 20, 30, 40, 50 };
06      int *pa = ary;                          // 첫 번째 배열 요소 주소
07      int *pb = pa + 3;                       // 네 번째 배열 요소 주소
08
09      printf("pa : %u\n", pa);
10      printf("pb : %u\n", pb);
```



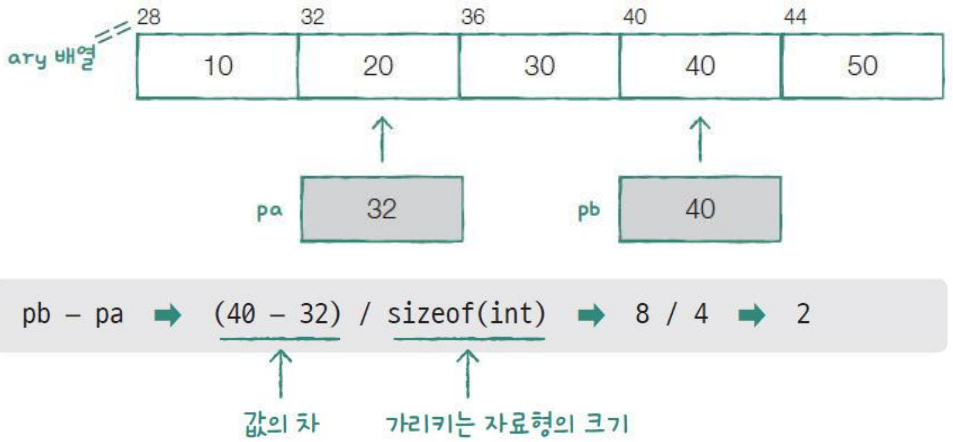
❖ 포인터의 뺄셈과 관계 연산 (2/2)

포인터의 뺄셈과 관계 연산 소스 코드 예제10-4.c

```
11      pa++;                                   // pa를 다음 배열 요소로 이동
12      printf("pb - pa : %u\n", pb - pa);      // 두 포인터의 뺄셈
13
14      printf("앞에 있는 배열 요소의 값 출력 : ");
15      if (pa < pb) printf("%d\n", *pa);       // pa가 배열의 앞에 있으면 *pa 출력
16      else printf("%d\n", *pb);               // pb가 배열의 앞에 있으면 *pb 출력
17
18      return 0;
19  }
```

실행결과

```
pa : 3799428
pb : 3799440
pb - pa : 2
앞에 있는 배열 요소의 값 출력 : 20
```



마무리

키워드로 끝내는 핵심 포인트

- **배열명**은 첫 번째 **요소의 주소**이다.
- **포인터에 배열명**을 저장하면 배열명처럼 사용할 수 있다.
- **배열명의 정수 덧셈**은 가리키는 자료형의 크기를 곱해서 더한다.
- **포인터의 뺄셈** 결과는 배열 요소 간의 간격 차이를 의미한다.

마무리

표로 정리하는 핵심 포인트

표 10-1 배열과 포인터

구분	사용 예	기능
배열명	`int ary[3];` `ary == &ary[0];`	배열명은 첫 번째 요소의 주소
배열명 + 정수	`int ary[3];` `ary + 1;`	가리키는 자료형의 크기를 곱해서 더한다. `ary + (1 * sizeof(*ary))`

배열명과 포인터는 같다.	`int ary[3];`
	`int *pa = ary;`
	`pa[1] = 10;`
		포인터가 배열명을 저장하면 배열명처럼 쓸 수 있다.
		두 번째 배열 요소에 10 대입

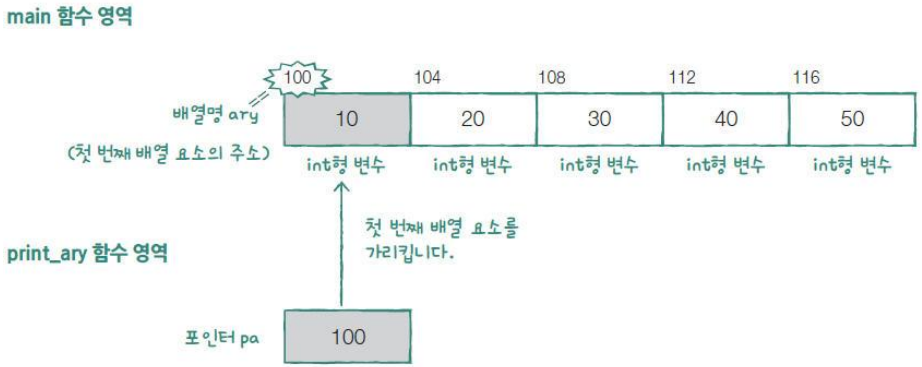
배열명과 포인터는 다르다.	`ary++; ( × )` `pa++; ( ○ )`	배열명은 상수이므로 그 값을 바꿀 수 없지만 포인터는 가능하다.

10-2 배열을 처리하는 함수

❖ 배열의 값을 출력하는 함수

배열을 처리하는 함수 소스 코드 예제10-5.c

```
01  #include <stdio.h>
02
03  void print_ary(int *pa);
04
05  int main(void)
06  {
07      int ary[5] = { 10, 20, 30, 40, 50 };
08
09      print_ary(ary);
10
11      return 0;
12  }
13
14  void print_ary(int *pa)
15  {
16      int i;
17
18      for (i = 0; i < 5; i++)
19      {
20          printf("%d ", pa[i]);
21      }
22  }
```



10-2 배열을 처리하는 함수

❖ 배열 요소의 개수가 다른 배열도 출력하는 함수

크기가 다른 배열을 출력하는 함수 소스 코드 예제10-6.c

```c
#include <stdio.h>

void print_ary(int *pa, int size);

int main(void)
{
    int ary1[5] = { 10, 20, 30, 40, 50 };
    int ary2[7] = { 10, 20, 30, 40, 50, 60, 70 };

    print_ary(ary1, 5);
    printf("\n");
    print_ary(ary2, 7);

    return 0;
}

void print_ary(int *pa, int size)
{
    int i;

    for (i = 0; i < size; i++)
    {
        printf("%d ", pa[i]);
    }
}
```

실행결과

```
10 20 30 40 50
10 20 30 40 50 60 70
```

10-2 배열을 처리하는 함수

❖ 배열에 값을 입력하는 함수 (1/2)

배열에 값을 입력하는 함수 소스 코드 예제10-7.c

```c
#include <stdio.h>

void input_ary(double *pa, int size);
double find_max(double *pa, int size);

int main(void)
{
    double ary[5];
    double max;
    int size = sizeof(ary) / sizeof(ary[0]);

    input_ary(ary, size);
    max = find_max(ary, size);
    printf("배열의 최댓값 : %.1lf\n", max);

    return 0;
}
```

실행결과

```
5개의 실수값 입력 : 3.4 0.5 1.7 5.2 2.0
배열의 최댓값 : 5.2
```

10- 2 배열을 처리하는 함수

❖ 배열에 값을 입력하는 함수 (2/2)

배열에 값을 입력하는 함수 소스 코드 예제10-7.c

```
19  void input_ary(double *pa, int size)
20  {
21      int i;
22
23      printf("%d개의 실수값 입력 : ", size);
24      for (i = 0; i < size; i++)
25      {
26          scanf("%lf", pa + i);
27      }
28  }
29
```

간접 참조 연산(*)
주소 연산(&)

pa + 2

*(pa + 2) 또는 pa[2]

&(*(pa + 2)) 또는 &pa[2]

```
30  double find_max(double *pa, int size)
31  {
32      double max;
33      int i;
34
35      max = pa[0];
36      for (i = 1; i < size; i++)
37      {
38          if (pa[i] > max) max = pa[i];
39      }
40
41      return max;     // 최댓값 반환
42  }
```

마무리

키워드로 끝내는 핵심 포인트

- **배열**을 **출력**하는 **함수**에 필요한 것은 배열명이다.
- **배열**에 **입력**하는 **함수**에 필요한 것도 배열명이다.
- 배열의 크기가 달라도 입출력이 가능하려면 **배열 요소의 개수**를 알아야 한다.

마무리

표로 정리하는 핵심 포인트

표 10-2 배열에 입출력하는 함수

	배열을 출력하는 함수	배열에 입력하는 함수
호출	`int ary[5] = { 10, 20, 30, 40, 50 };` `print_ary(ary, 5);`	`int ary[5];` `input_ary(ary, 5);`

정의	`void print_ary(int *pa, int size)`
	`{`
	`  int i;`
	`  for (i = 0; i < size; i++)`
	`  {`
	`  printf("%d ", pa[i]);`
	`  }`
	`}`
		`void input_ary(int *pa, int size)`
		`{`
		`  int i;`
		`  for (i = 0; i < size; i++)`
		`  {`
		`  scanf("%d", pa + i);`
		`  }`
		`}`