시험에 나오는 것만 공부한다! 시나공

두 번 시험보면 한 번은 출제되는 프로그래밍 – 포인터 5문제

정보처리기사 실기

1. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
#include <stdio.h>
int len(char* p);

int main( ) {
    char* p1 = "2022";
    char* p2 = "202207";
    int a = len(p1);
    int b = len(p2);
    printf("%d", a + b);
}

int len(char* p) {
    int r = 0;
    while (*p != '\0') {
        p++;
        r++;
    }
    return r;
}
```

답 : 10

[해설]

```
#include <stdio.h>
int len(char* p);          len( ) 함수의 프로토타입 선언이다.
main( ) {
❶   char* p1 = "2022";
❷   char* p2 = "202207";
❸⑩ int a = len(p1);
⑪⑱ int b = len(p2);
⑲   printf("%d", a + b);
}
❹⑫ int len(char* p) {
❺⑬     int r = 0;
```

```
❻⓮     while (*p != '\0') {
❼⓯         p++;
❽⓰         r++;
        }
❾⓱     return r;
    }
```

모든 C 프로그램은 반드시 main( ) 함수에서 시작한다.

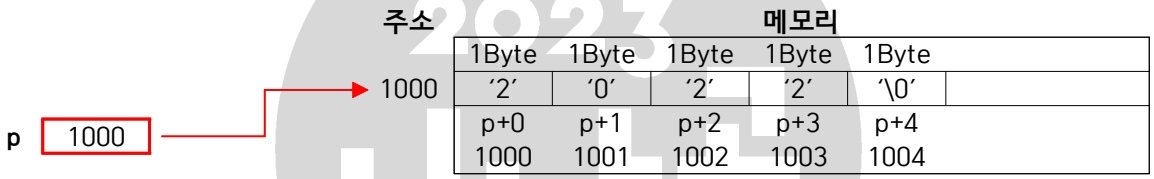
❶ 문자형 포인터 변수 p1을 선언하고 “2022”를 가리키는 주소로 초기화한다.

❷ 문자형 포인터 변수 p2를 선언하고 “202207”을 가리키는 주소로 초기화한다. 다음의 그림에서 p1과 p2가 할당된 공간의 주소는 임의로 정한 것이며, 이해를 돕기 위해 10진수로 표현했다.

	주소	메모리						
p1	1000	‘2’	‘0’	‘2’	‘2’	‘\0’		
p2	2000	‘2’	‘0’	‘2’	‘2’	‘0’	‘7’	‘\0’

❸ 정수형 변수 a를 선언하고 p1의 값을 인수로 len( ) 함수를 호출한 후 돌려받은 값으로 초기화한다.

❹ 정수를 반환하는 len( ) 함수의 시작점이다. ❸번에서 전달받은 p1의 값을 문자형 포인터 변수 p가 받는다.



❺ 정수형 변수 r을 선언하고 0으로 초기화한다. r은 문자의 수를 카운트하기 위한 변수이다.

❻ p가 가리키는 곳의 값이 ‘\0’이 아닌 동안 ❼, ❽번을 반복 수행한다.

❼ ‘p = p + 1;’과 동일하다. ‘\0’이 나올 때까지, 즉 문자열의 끝을 찾을 때까지 주소를 1씩 증가시킨다.

❽ ‘r = r + 1;’과 동일하다. “2022”에 포함된 문자의 개수를 센다.

반복문 실행에 따른 변수들의 변화는 다음과 같다.

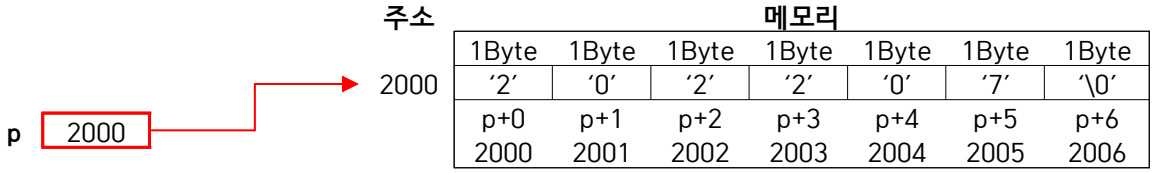
p	r	*p
1000	0	‘2’
1001	1	‘0’
1002	2	‘2’
1003	3	‘2’
1004	4	‘\0’

❾ 함수를 호출했던 ❿번으로 r의 값, 즉 문자의 개수 4를 반환한다.

❿ a에는 4가 저장된다.

⓫ 정수형 변수 b를 선언하고 p2의 값을 인수로 len( ) 함수를 호출한 후 돌려받은 값으로 초기화한다.

⓬ 정수를 반환하는 len( ) 함수의 시작점이다. ⓫번에서 전달받은 p2의 값을 p가 받는다.



⓭ 정수형 변수 r을 선언하고 0으로 초기화한다.
⓮ p가 가리키는 곳의 값이 '\0'이 아닌 동안 ⓯, ⓰번을 반복 수행한다.
⓯ 'p = p + 1;'과 동일하다. '\0'이 나올 때까지 문자열이 저장된 주소를 1씩 증가시킨다.
⓰ 'r = r + 1;'과 동일하다. "202207"에 포함된 문자의 개수를 센다.
반복문 실행에 따른 변수들의 변화는 다음과 같다.

p	r	*p
2000	0	'2'
2001	1	'0'
2002	2	'2'
2003	3	'2'
2004	4	'0'
2005	5	'7'
2006	6	'\0'

⓱ 함수를 호출했던 ⓲번으로 r의 값 6을 반환한다.
⓲ b에는 6이 저장된다.
⓳ 4+6의 결과인 10을 정수로 출력한다.

결과 10

2. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
#include <stdio.h>
int main( ) {
    int a[4] = { 0, 2, 4, 8 };
    int b[3];
    int* p;
    int sum = 0;
    for (int i = 1; i < 4; i++) {
        p = a + i;
        b[i - 1] = *p - a[i - 1];
        sum = sum + b[i - 1] + a[i];
    }
    printf("%d", sum);
}
```

답 : 22

[해설]

```
#include <stdio.h>
int main() {
❶   int a[4] = { 0, 2, 4, 8 };
❷   int b[3];
❸   int* p;
❹   int sum = 0;
❺   for (int i = 1; i < 4; i++) {
❻       p = a + i;
❼       b[i - 1] = *p - a[i - 1];
❽       sum = sum + b[i - 1] + a[i];
    }
❾   printf("%d", sum);
}
```

❶ 4개의 요소를 갖는 정수형 배열 a를 선언하고 초기화한다.

	[0]	[1]	[2]	[3]
a	0	2	4	8

❷ 3개의 요소를 갖는 정수형 배열 b를 선언한다.

	[0]	[1]	[2]
b			

❸ 정수형 포인터 변수 p를 선언한다.

❹ 정수형 변수 sum을 선언하고 0으로 초기화한다.

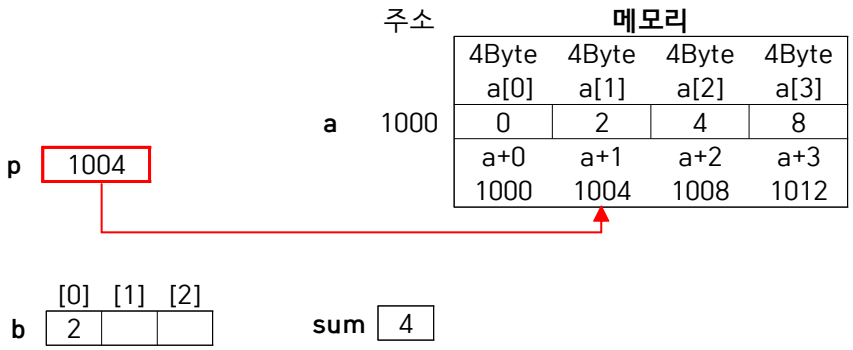
❺ 반복 변수 i가 1부터 1씩 증가하면서 4보다 작은 동안 ❻~❽번을 반복 수행한다.

첫 번째 반복 (i = 1)

❻ p에 a+1의 주소를 저장한다. p에 a 배열의 두 번째 요소인 a[1]의 주소를 저장한다.

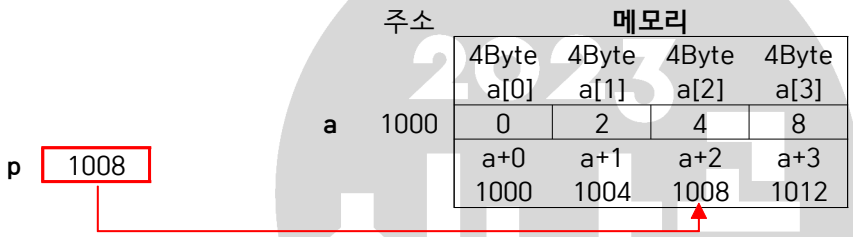
❼ b[0]에 p가 가리키는 곳의 값 2에서 a[0]의 값 0을 뺀 2를 저장한다.

❽ sum에 b[0]의 값 2와 a[1]의 값 2를 더한 값 4를 누적한다.



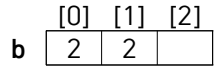
두 번째 반복 (i = 2)

• p에 a+2의 주소인 1008을 저장한다.



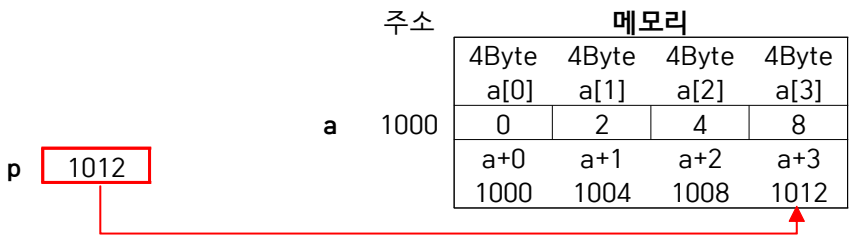
• b[1]에 p가 가리키는 곳의 값 4에서 a[1]의 값 2를 뺀 값인 2를 저장한다.

• sum에 b[1]의 값 2와 a[2]의 값 4를 더한 값 6을 누적한다.



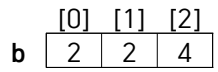
세 번째 반복 (i = 3)

• p에 a+3의 주소인 1012를 저장한다.



• b[2]에 p가 가리키는 곳의 값 8에서 a[2]의 값 4를 뺀 값인 4를 저장한다.

• sum에 b[2]의 값 4와 a[3]의 값 8을 더한 값 12를 누적한다.



• i가 4가 되면서 for문을 빠져나가 ❾번으로 이동한다.

❾ sum의 값 22를 정수로 출력한다.

결과 22

3. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

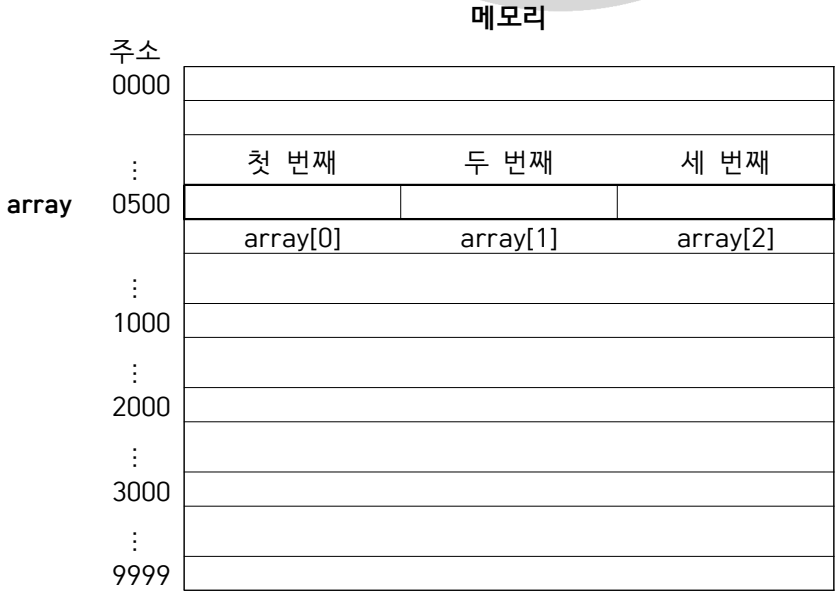
```
#include <stdio.h>
int main() {
    int* array[3];
    int a = 12, b = 24, c = 36;
    array[0] = &a;
    array[1] = &b;
    array[2] = &c;
    printf("%d", *array[1] + **array + 1);
}
```

답 : 37

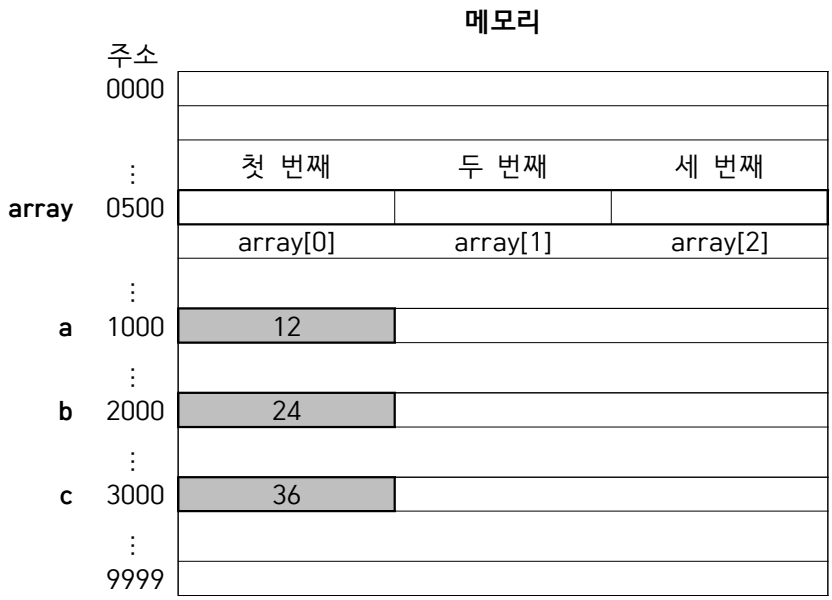
[해설]

```
#include <stdio.h>
int main() {
❶    int* array[3];
❷    int a = 12, b = 24, c = 36;
❸    array[0] = &a;
❹    array[1] = &b;
❺    array[2] = &c;
❻    printf("%d", *array[1] + **array + 1);
}
```

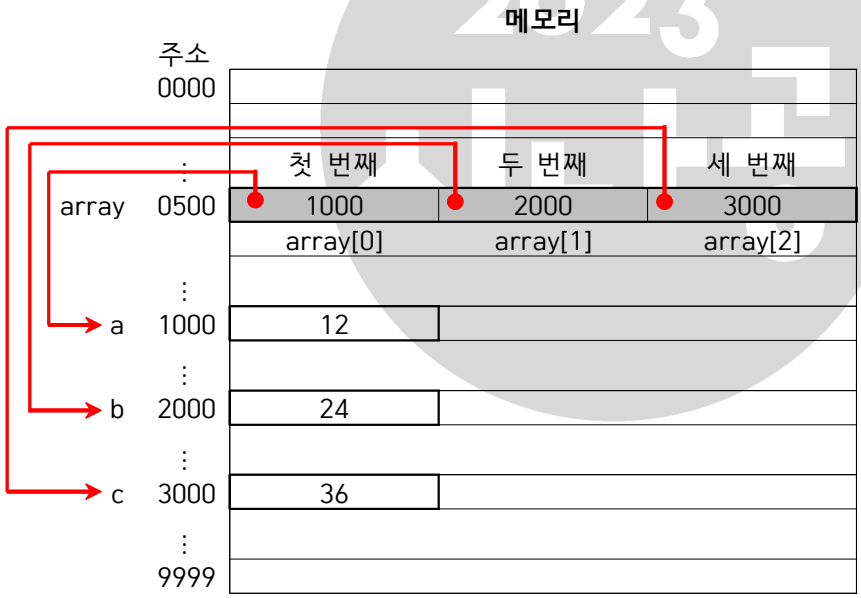
❶ 3개의 요소를 갖는 정수형 포인터 배열 array를 선언한다. 주소는 임의로 정한 것이다.



❷ 정수형 변수 a, b, c에 각각 12, 24, 36을 저장한다.



❸ array[0]에 a의 주소를 저장한다.
❹ array[1]에 b의 주소를 저장한다.
❺ array[2]에 c의 주소를 저장한다.



❻ array[1]이 가리키는 곳의 값과 *array가 가리키는 곳의 값과 1을 더한 후 정수로 출력한다.

- ***array[1]** : array[1]에는 2000이 저장되어 있고 2000이 가리키는 곳의 값은 24이다.
- ****array**
  - array : 배열의 이름만 지정하면 배열의 첫 번째 요소의 주소인 &array[0], 즉 500을 의미한다.
  - *array : array는 500이고 500이 가리키는 곳의 값은 1000이다.
  - **array : *array는 1000이고 1000이 가리키는 곳의 값은 12이다.

∴ 24 + 12 + 1 = 37

결과 37

4. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

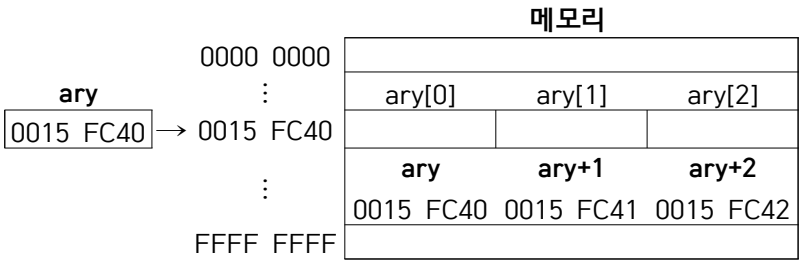
```
#include <stdio.h>
int main() {
	int ary[3];
	int s = 0;
	*(ary + 0) = 1;
	ary[1] = *(ary + 0) + 2;
	ary[2] = *ary + 3;
	for (int i = 0; i < 3; i++)
		s = s + ary[i];
	printf("%d", s);
}
```

답 : 8

[해설]

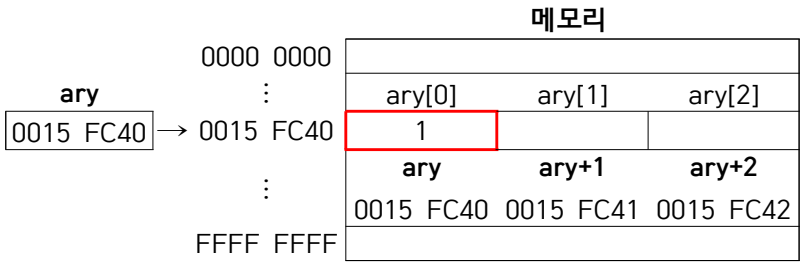
```
#include <stdio.h>
int main() {
❶	int ary[3];
❷	int s = 0;
❸	*(ary + 0) = 1;
❹	ary[1] = *(ary + 0) + 2;
❺	ary[2] = *ary + 3;
❻	for (int i = 0; i < 3; i++)
❼		s = s + ary[i];
❽	printf("%d", s);
}
```

❶ 3개의 요소를 갖는 정수형 배열 ary를 선언한다.

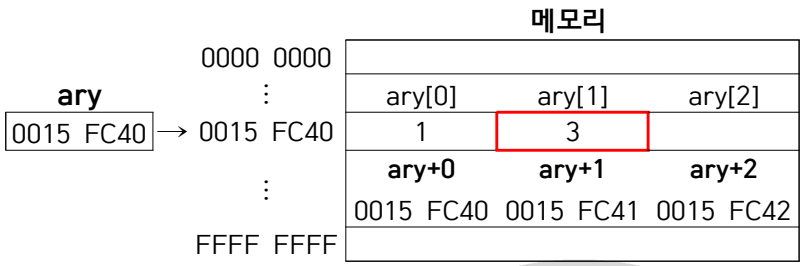


❷ 정수형 변수 s를 선언하고 0으로 초기화한다.

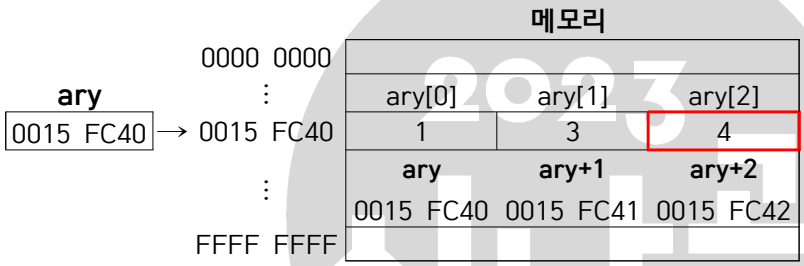
❸ ary+0이 가리키는 곳에 1을 저장한다.



❹ ary[1]에 ary+0이 가리키는 곳의 값 1에 2를 더한 값을 저장한다.



❺ ary[2]에 ary가 가리키는 곳의 값 1에 3을 더한 값을 저장한다.



❻ 반복 변수 i가 0에서 시작하여 1씩 증가하면서 3보다 작은 동안 ❼번을 반복 수행한다.

❼ s에 ary[i]의 값을 누적한다.

반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	ary[i]	s
0	1	1
1	3	4
2	4	8
3		

❽ s의 값을 출력한다.

결과 8

5. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)

```
#include <stdio.h>
main() {
        char* p = "KOREA";
        printf("%s\n", p);
        printf("%s\n", p + 3);
        printf("%c\n", *p);
        printf("%c\n", *(p + 3));
        printf("%c\n", *p + 2);
}
```

답 :

KOREA

EA

K

E

M

[해설]

```
#include <stdio.h>
main() {
❶       char* p = "KOREA";
❷       printf("%s\n", p);
❸       printf("%s\n", p + 3);
❹       printf("%c\n", *p);
❺       printf("%c\n", *(p + 3));
❻       printf("%c\n", *p + 2);
}
```

❶ 문자형 포인터 변수 p를 선언하고, 문자열 "KOREA"가 저장된 곳의 주소를 저장한다.

메모리

p		1Byte	1Byte	1Byte	1Byte	1Byte	1Byte
0015 FC40 →	0000 0000 ⋮ 0015 FC40 ⋮ FFFF FFFF	'K'	'O'	'R'	'E'	'A'	'\0'
		p	**p+1**	**p+2**	**p+3**	**p+4**	**p+5**
		0015 FC40	0015 FC41	0015 FC42	0015 FC43	0015 FC44	0015 FC45

※ 문자열을 저장하는 경우 문자열의 끝을 의미하는 널 문자('\0')가 추가로 저장되며, 출력시 널 문자는 표시되지 않습니다.

❷ p의 위치부터 문자열의 끝('\0')까지 모든 문자를 하나의 문자열로 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과 KOREA

❸ p+3의 위치부터 문자열의 끝('\0')까지 모든 문자를 하나의 문자열로 출력하고 커서를 다음 줄의 처

음으로 옮긴다.

결과
```
KOREA
EA
```

❹ p가 가리키는 곳의 문자를 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과
```
KOREA
EA
K
```

❺ p+3이 가리키는 곳의 문자를 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과
```
KOREA
EA
K
E
```

❻ p가 가리키는 곳의 문자에 2를 더한 값을 출력하고 커서를 다음 줄의 처음으로 옮긴다.

※ “KOREA”라는 문자열이 메모리에 저장될 때 문자로 저장되는 것이 아니라 해당 문자의 아스키 코드 값이 저장됩니다. 즉 ‘K’는 ‘K’에 해당하는 아스키 코드 값인 75가 저장됩니다. 그러므로 p가 가리키는 곳의 값인 75에 2를 더한 77을 문자로 출력한다는 것은 알파벳 순서상 ‘K’의 다다음 문자인 ‘M’을 출력한다는 의미입니다.

결과
```
KOREA
EA
K
E
M
```