비기전수 C프로그래밍 정오표

|  |  |  |  |
| --- | --- | --- | --- |
| **장** | **Page** | **수정 전** | **수정 후** |
| Chapter.3  3-3 | 53쪽  그림 3-7 | 53 바탕.tif |  |
| Chapter.3  3-4 | 57쪽  그림 3-12 | 57-1.bmp |  |
| Chapter.3  3-5 | 63쪽 가운데 | → 부동소수점에 오차가 존재한다는 사실!  → 부동소수점에 오차가 존재하는 이유! | → 부동소수점 오차가 존재한다는 사실!  → 부동소수점 오차가 존재하는 이유! |
| Chapter.8  8-1 | 153쪽  그림 8-2  오른쪽 그림 | → Printf("%d", & n1); | → Printf("%d", n1); |
| Chapter.8  8-3 | 155쪽  그림 8-3  오른쪽 그림 | → Printf("%f", & n1); | → Printf("%f", n2); |
| Chapter.8  8-2 | 156쪽 가운데 | 이 예제의 9행과 13행에서 scanf 함수에 전달되는 문자열은 각각 다음과 같다.  “%d %d %”  “%1f %1f %1f” | 이 예제의 9행과 13행에서 scanf 함수에 전달되는 문자열은 각각 다음과 같다.  “%d %d d”  “%1f %1f %1f” |
| Chapter.11  11-2 | 229쪽  예제 11-1.c  5행 | 5 char ch’1 = ‘A’; | 5 char ch1 = ‘A’; |
| Chapter.11  11-3 | 232쪽  예제 11-2.c  6행 | 4 int main(void)  5 {  6 char ch = ‘0’;  7 | 4 int main(void)  5 {  6 char ch = ‘0’;  7 |
| Chapter.13  13-1 | 254쪽  표 13-1  2행 | |  |  |  | | --- | --- | --- | | %u | Unsigned in t | 부호 없는 10진수 정수 | | |  |  |  | | --- | --- | --- | | %u | Unsigned int | 부호 없는 10진수 정수 | |
| Chapter.13  13-1 | 258쪽  예제 13-3.c  13행 | 13 printf(“%g ＼n”, d4); // 1.23E-006 출력 | 13 printf(“%g ＼n”, d4); //1.23e-006 출력 |
| Chapter.16  16-2 | 296쪽 가운데 | 이 두 문장을 실제 코드로 옮기면 다음과 같다.  arr[0]=10; // 첫 번째 배열 요소에 10 저장  arr[1]=20; // 두 번째 배열 요소에 20 저장 | 이 두 문장을 실제 코드로 옮기면 다음과 같다.  array[0]=10; // 첫 번째 배열 요소에 10 저장  array[1]=20; // 두 번째 배열 요소에 20 저장 |
| Chapter.19  19-1 | 354쪽 가운데 | 앞에서 문자열의 전달 방식을 질문했던 후배가 여러분을 찾아와 예제 20-4를 보면서 질문한다고 가정하자. | 앞에서 문자열의 전달 방식을 질문했던 후배가 여러분을 찾아와 예제 19-4를 보면서 질문한다고 가정하자. |
| Chapter.19  19-1 | 355쪽 아래 | 안타깝지만 함수 안에서는 인자로 전달된 배열의 주소 값만 가지고, 배열의 크기를 계산할 수 없다. 따라서 함수의 주소 값을 인자로 전달할 때에는 배열의 길이도 미리 계산해서 더불어 전달해야 한다. 즉 위의 코드는 다음과 같이 변경되어야 한다. 후배에게 전달해 준 코드는 아래의 코드가 되었어야 했다. | 안타깝지만 함수 안에서는 인자로 전달된 배열의 주소 값만 가지고, 배열의 크기를 계산할 수 없다. 따라서 함수의 인자로 배열의 주소 값을 전달할 때에는 배열의 길이도 미리 계산해서 더불어 전달해야 한다. 즉 위의 코드는 다음과 같이 변경되어야 한다. 후배에게 전달해 준 코드는 아래의 코드가 되었어야 했다. |
| Chapter.22  22-1 | 411쪽 위에 | \*ptr1 = 100  \*ptr2 = 200 | \*ptr1 = 200  \*ptr2 = 100 |
| Chapter.24  24-3 | 453쪽 가운데 | #include<stdlib.h>  void \* calloc(size\_t elt\_count, elt\_size)  성공 시 할당된 메모리의 주소 값, 실패 시 NULL 반환 | #include<stdlib.h>  void \* calloc(size\_t elt\_count, size\_t elt\_size)  성공 시 할당된 메모리의 주소 값, 실패시 NULL 반환 |
| Chapter.27  27-3 | 504쪽 가운데 | → 13행: 함수의 반환형이 void임에도 불구하고 return문이 삽입되었다. “아니 어떻게 반환형이 void인데 return문이 삽입될 수 있지?” 물론 값의 변환이라는 의미를 담으면 삽입이 불가능하다. 하지만 13행의 return문은 값을 반환하지 않는다(반환할 대상이 명시되어 있지 않다). 즉 키워드 return만 존재하여 함수를 종료시키기 위한 용도로 사용되었기 때문에 문제가 되지 않는다.  → 13행, 14행: n이 0이면 12행의 if문이 참이 되어, 13행에 의해서 함수를 빠져나오게 된다. 즉 탈출조건과 그에 따른 탈출문이 삽입되었다.  → 16행: 재귀적으로 Recursive 함수가 호출되는 부분이다. 그런데 매개변수 n에 저장된 값보다 하나 작은 값을 전달하고 있다. 따라서 언젠가는 0이 전달되어 13행의 if 조건문이 참이 될 것이다. | → 14행: 함수의 반환형이 void임에도 불구하고 return문이 삽입되었다. “아니 어떻게 반환형이 void인데 return문이 삽입될 수 있지?” 물론 값의 반환이라는 의미를 담으면 삽입이 불가능하다. 하지만 14행의 return문은 값을 반환하지 않는다(반환할 대상이 명시되어 있지 않다). 즉 키워드 return만 존재하여 함수를 종료시키기 위한 용도로 사용되었기 때문에 문제가 되지 않는다.  → 13행, 14행: n이 1이면 13행의 if문이 참이 되어, 14행에 의해서 함수를 빠져나오게 된다. 즉 탈출조건과 그에 따른 탈출문이 삽입되었다.  → 16행: 재귀적으로 Recursive 함수가 호출되는 부분이다. 그런데 매개변수 n에 저장된 값보다 하나 작은 값을 전달하고 있다. 따라서 언젠가는 1이 전달되어 13행의 if 조건문이 참이 될 것이다. |
| Chapter.27  27-3 | 506쪽  그림 27-7 | 507.tif |  |
| Chapter.27  27-3 | 507쪽 위 | → 19, 20행: 그림 6-15의 식에서 n이 1인 경우를 코드로 옮겨 놓은 것이다.  → 21, 22행: 그림 6-15의 식에서 n이 2 이상인 경우를 코드로 옮겨 놓은 것이다. | → 19, 20행: 그림 27-7의 식에서 n이 1인 경우를 코드로 옮겨 놓은 것이다.  → 21, 22행: 그림 27-7의 식에서 n이 2 이상인 경우를 코드로 옮겨 놓은 것이다. |