《C 포인터의 이해와 활용》 1쇄본 오탈자

쪽	쪽 내 위치	수정 전	수정 후
3	위에서 둘째줄	함수가 호출되는 동안에만 존재한다.	함수가 호출되는 동안에만 존재한다. 일반적으로 블록문 안에서 선언된 변수의 범위는 해당 블록으로 제한된다. 이들은 종종 자동 변수라고 언급된다.
5	위에서 6번째줄	하지만 최신 C 표준인 C11에서 이미 가변 배열을 지원하고 있다.	하지만 가변 크기 배열은 C99부터 이미 지원하고 있다.
14	가운데	printf("%p₩n",*pi); // 5 출력	printf("%d₩n",*pi); // 5 출력
18	밑에서 5번째 (두 번째 불릿기호)	void 포인터의 실제 동작은 시스템에 의존적이다.	void 포인터의 실제 동작은 시스템에 따라 다르다.
22	세 번째 불릿기호	intptr_t와 uintprt_t	intptr_t와 uintptr_t
28	표 1-5	byte 1	(삭제, C에는 byte 타입이 없습니다.)
30	밑에서 5번째줄	대부분의 컴파일러는 void 타입을 1바이트 크기의 데이터 타입으로 다룬다. pv가 void 타입에 대한 포인터이므로 주소가 1바이트만큼 증가한다.	대부분의 컴파일러는 void 데이터 타입을 마치 한 바이트의 크기를 가지는 것처럼 처리한다. 포인터가 void를 가리킬 경우 해당 포인터의 주소는 1씩 증가한다.
34	그림 1-10	title	(그림 안의 title들을 titles로 수정합니다.)
44	밑에서 셋째줄	C에서 동적 메모리 할당의 기본 단계는 다음과 같다.	C에서 동적 메모리 할당의 기본 단계는 다음과 같다. 동적 메모리 할당과 관련된 함수들은 stdlib.h 헤더 파일에서 찾을 수 있다.
47	코드 3번째줄	*pc[i] = 0;	pc[i] = 0;
57	마지막줄	인자에 따라 realloc 함수가 어떻게 동작하는지 표 2-2에 정리하였다.	인자에 따라 realloc 함수가 어떻게 동작하는지 표 2-2에 정리하였다. realloc 함수의 실제 동작은 구현에 따라 달라질 수 있다는 것을 명심하도록 하자. 사용하기 전에 항상 동작을 확인하도록 하자.

최종수정일자: 2016-04-11

《C 포인터의 이해와 활용》 1쇄본 오탈자

쪽	쪽 내 위치	수정 전	수정 후
100	첫 번째 코드 3번째줄	operations['-'] = subtract;	operations['-'] = sub;
102	마지막 코드 세번째줄	basePointer = (fptrToSingleInt)fptrFirst;	basePointer = (fptrBase)fptrFirst;
118	코드에서 밑에서 4번째줄	char* buffer = (char*)malloc(strlen(" cat")+1);	char* word = (char*)malloc(strlen(" cat")+1);
118	코드에서 밑에서 3번째줄	strcpy(buffer," cat");	strcpy(word," cat");
118	코드에서 밑에서 2번째줄	printf("%s₩n",trim(buffer));	printf("%s₩n",trim(word));
130	코드 첫번째 줄	printf("%d ",(arr+i) [j]);	printf("%d ",(arr+i*cols) [j]);
131	위에서 4번째줄	arr3d[1][0]는 배열의 두 번째 열, 첫 번째 행의 요소를 가리키는데, 이것은 크기가 5인 1차원 배열에 대한 포인터다.	arr3d[1][0]는 배열의 두 번째 열, 첫 번째 행의 요소를 가리키는데, 이것은 크기가 16인 1차원 배열에 대한 포인터다.

(계속)

최종수정일자: 2016-04-11

《C 포인터의 이해와 활용》 1쇄본 오탈자

쪽	쪽 내 위치	수정 전	수정 후

137	가운데 코드	int row = 0;
		for(int i=0; i<4; i++) {
		printf("layer1[%d][%d] Address: %p Value: %d₩n",
		row, i, &arr2[row][i], arr2[row][i]);
		}
		printf("₩n");
		
		row = 1;
		for(int i=0; i<2; i++) {
		printf("layer1[%d][%d] Address: %p Value: %d₩n",
		row, i, &arr2[row][i], arr2[row][i]);
		}
		printf("₩n");
		
		row = 2;
		for(int i=0; i<3; i++) {
		printf("layer1[%d][%d] Address: %p Value: %d₩n",
		row, i, &arr2[row][i], arr2[row][i]);
		}
		printf("₩n");
			int row = 0;
			for(int i=0; i<4; i++) {
			printf("arr2[%d][%d] Address: %p Value: %d₩n",
			row, i, &arr2[row][i], arr2[row][i]);
			}
			printf("₩n");
			
			row = 1;
			for(int i=0; i<2; i++) {
			printf("arr2[%d][%d] Address: %p Value: %d₩n",
			row, i, &arr2[row][i], arr2[row][i]);
			}
			printf("₩n");
			
			row = 2;
			for(int i=0; i<3; i++) {
			printf("arr2[%d][%d] Address: %p Value: %d₩n",
			row, i, &arr2[row][i], arr2[row][i]);
			}
			printf("₩n");

161	밑에서 5번째줄	이는 중복이므로 불필요하다. 게다가, 종종 컴파일 경고를 발생시킨다	이 코드는 정상적으로 동작하지만, char에 대한 포인터 대신 char에 대한 포인터의 포인터가 전달되므로 경고 메시지가 출력된다.
164	5번째줄	format 함수의 사용 방식은 다음과 같다.	다음 코드는 format 함수의 사용법을 보여주며, 여기서 사용된 buffer는 배열로 선언되었다고 가정한다. 만약 buffer가 동적으로 할당된 메모리인 경우, 버퍼의 크기 값으로 sizeof 함수 대신에 할당 시에 사용된 크기를 전
168	코드 2~5줄	static char* bpCenter = "Boston Processing Center"; static char* dpCenter = "Denver Processing Center"; static char* apCenter = "Atlanta Processing Center"; static char* sjpCenter = "San Jose Processing Center";	static char bpCenter[] = "Boston Processing Center"; static char dpCenter[] = "Denver Processing Center"; static char apCenter[] = "Atlanta Processing Center"; static char sjpCenter[] = "San Jose Processing Center";
186	밑에서 7번째줄	반환 값이 -1이면 첫 번째 employee가 두 번째 employee보다 사전 적으로 앞에 위치함을 뜻한다. 반환 값이 1이면 첫 번째 employee가 두 번째 employee보 다 사전적으로 뒤에 위치함을 뜻한다.	반환 값이 음수이면 첫 번째 employee가 두 번째 employee보다 사전적으로 앞에 위치함을 뜻한다. 반환 값이 양수이면 첫 번째 employee가 두 번째 employee보다 사전적으로 뒤에 위치함을 뜻한다.

최종수정일자: 2016-04-11

《C 포인터의 이해와 활용》 1쇄본 오탈자

쪽	쪽 내 위치	수정 전	수정 후
186	밑에서 4번째줄	DISPLAY 함수 포인터는 void*형의 매개변수를 입력받고, 아무 값도 반환하지 않는(void) 함수를 지칭한다.	DISPLAY 함수 포인터는 void에 대한 포인터를 전달하고, void를 반환하는 함수를 지칭한다.

226	두번째 코드	char* name = (char*)malloc(...);
		...
		memset(name,0,sizeof(name));
		free(name);
			int SIZEOFFIELD = ...;
			char* name = (char*)malloc(SIZEOFFIELD);
			...
			memset(name,0,SIZEOFFIELD);
			free(name);

226	밑에서 2번째줄	이 장에서 다른 이슈 대부분을	이 장에서 다룬 이슈 대부분을
226	밑에서 첫번째줄	예를 들어, GCC 컴파일러의 2Wall 옵션은 모든 컴파일러 경고를 출력하도록 한다.	예를 들어, GCC 컴파일러의 -Wall 옵션을 사용하면 많은 컴파일 경고가 출력되도록 할 수 있지만, 모든 컴파일러 경고가 출력되는 것은 아니다. 자세한 컴파일러 옵션에 대해서는 컴파일러의 문서를 확인하도록 하자.
237	두번째 코드 (발바닥) 4번째줄	int volatile ptr3 = #	int volatile *ptr3 = #
244	세번째 코드 2~3줄	VectorInfo *vectorInfo = Product->info; int beginningIndex = Product->beginningIndex;	VectorInfo *vectorInfo = product->info; int beginningIndex = product->beginningIndex;
254	5,9번째줄	vFunction	vFunctionS

257	코드 6번째줄
	(빈 행 포함)
		void rectangleSetY(Rectangle *rectangle, int y) {
		rectangle->base.y;
		}
			void rectangleSetY(Rectangle *rectangle, int y) {
			rectangle->base.y = y;
			}

최종수정일자: 2016-04-11