C言語: ポインタとconstの完全要約メモ

1. constの基本ルール

宣言	意味	例	書き換え可否
const int a	aの値を変更できない	a = 10;	
int* p	intを指すポインタ	*p = 5;	
const int* p	読み取り専用intを指すポインタ	*p = 5;	
int* const p	固定ポインタ（指す先を変えない）	p = &x;	
const int* const p	読み取り専用値かつ固定ポインタ	—	

2. ポインタ配列と関数引数の型変換

宣言	配列の中身	関数引数での型
int arr[]	値（int）	int*
char arr[][10]	文字列（固定長）	char (*)[10]
const char* arr[]	文字列リテラルへのポインタ	const char**

3. qsort関数の比較関数のポイント

説明	内容
a, b	配列要素のアドレス（const void*）
キャスト	(const char* const*)a; (const char* const*)b;
意味	読み取り専用ポインタを指すポインタ

4. void* と const void* の変換ルール

変換	許可される？	理由
任意型 → void*	OK	どんな型でも受け入れ可能
void* → 任意型	OK	元に戻すのも安全
任意型 → const void*	OK	読み取り専用に変換は安全
const void* → 非const型	NG	constを外すのは危険

5. constの段階的な意味（例：二重ポインタ）

型	読み方	変更できないもの
const char**	文字列の内容は変更不可	文字列の中身
const char* const*	文字列もポインタも変更不可	文字列と指し先

6. よくある混乱の整理

状況	正しい型	解説
文字列リテラル配列をソート	const char* arr[]	並び替えOK・内容変更NG
比較関数内で文字列を参照	const char* const*	読み取り専用で安全
voidポインタの変換	(const char* const*)a	明示キャストで意図を明確に

7. constの位置ルール

ルール	意味
* の左に const	値を守る
* の右に const	ポインタを守る
両方に const	両方守る