sequence of multibyte characters; `%ls` is used to read a string of multibyte characters (a null character is added at the end). The `%l` [*set*] and `%l` [`^`*set*] conversion specifications can also read a string of multibyte characters.

`scanf` Examples

The next three tables contain sample calls of `scanf`. Each call is applied to the input characters shown to its right. Characters printed in ~~strikeout~~ are consumed by the call. The values of the variables after the call appear to the right of the input.

The examples in Table 22.14 show the effect of combining conversion specifications, white-space characters, and non-white-space characters. In three cases no value is assigned to `j`, so it retains its value from before the call of `scanf`. The examples in Table 22.15 show the effect of assignment suppression and specifying a field width. The examples in Table 22.16 illustrate the more esoteric conversion specifiers (`i`, `[`, and `n`).

Table 22.14 `scanf` Examples (Group 1)

`scanf` *Call*	*Input*	*Variables*
`n = scanf("%d%d", &i, &j);`	~~`12`~~•, •`34`␣	n: 1 i: 12 j: unchanged
`n = scanf("%d,%d", &i, &j);`	~~`12`~~•, •`34`␣	n: 1 i: 12 j: unchanged
`n = scanf("%d ,%d", &i, &j);`	~~`12`•, •`34`␣~~	n: 2 i: 12 j: 34
`n = scanf("%d, %d", &i, &j);`	~~`12`•, •`34`␣~~	n: 1 i: 12 j: unchanged

Table 22.15 `scanf` Examples (Group 2)

`scanf` *Call*	*Input*	*Variables*
`n = scanf("%*d%d", &i);`	~~`12`•`34`␣~~	n: 1 i: 34
`n = scanf("%*s%s", str);`	~~`My`•~~~~`Fair`~~•`Lady`␣	n: 1 str: "Fair"
`n = scanf("%1d%2d%3d", &i, &j, &k);`	~~`12345`␣~~	n: 3 i: 1 j: 23 k: 45
`n = scanf("%2d%2s%2d", &i, str, &j);`	~~`123456`␣~~	n: 3 i: 12 str: "34" j: 56