Recent Changes to C

November 15, 1978

A few extensions have been made to the C language beyond what is described in the reference document (‘‘The C Programming Language,’’ Prentice-Hall, 1978).

1. Structure assignment

Structures may be assigned, passed as arguments to functions, and returned by functions. The types of the operands taking part must be the same. Other plausible operators, such as equality comparison, have not been implemented.

There is a defect in the PDP-11 implementation of functions that return structures: if an interrupt occurs during the return sequence, and the same function is called reentrantly during the interrupt, the value returned from the first call may be corrupted. The problem can occur only in the presence of true interrupts, as in an operating system or a user program that makes significant use of signals; ordinary recursive calls are quite safe.

2. Enumeration type

There is a new data type analogous to the scalar types of Pascal. To the type-specifiers in the syntax on p. 193 of the C book add

enum-specifier

with syntax

enum-specifier:
`enum {` *enum-list* `}`
`enum` *identifier* `W {` *enum-list* `W}`
`enum` *identifier*

enum-list:
enumerator
enum-list `,` *enumerator*

enumerator:
identifier
identifier = *constant-expression*

The role of the identifier in the enum-specifier is entirely analogous to that of the structure tag on a struct-specifier; it names a particular enumeration. For example,

```
enum color { chartreuse, burgundy, claret, winedark };
. . .
enum color *cp, col;
```

makes `color` the enumeration-tag of a type describing various colors, and then declares `cp` as a pointer to an object of that type, and `col` as an object of that type.

The identifiers in the enum-list are declared as constants, and may appear wherever constants are required. If no enumerators with = appear, then the values of the constants begin at 0 and increase by 1 as the declaration is read from left to right. An enumerator with = gives the associated identifier the value indicated; subsequent identifiers continue the progression from the assigned value.

Enumeration tags and constants must all be distinct, and unlike structure tags and members, are drawn from the same set as ordinary identifiers.

Objects of a given enumeration type are regarded as having a type distinct from objects of all other types, and *lint* flags type mismatches. In the PDP-11 implementation all enumeration variables are treated as if they were `int`.