

```

hash_table: proc;
dcl ((n, n_min, m, mult) fixed,
      (lim1, lim2) float,
      p ptr) static;
dcl 1 table(n) ctl(p),
      2 (full, skip) bit(1),
      2 (key, value) fixed;

      /* initialize table */

```

```

initialize: proc;
  dcl i fixed;
  m = 0;
  allocate table set(p);
  do i = 1 to n;
    p -> table(i).full, p -> table(i).skip = "0"b;
  end;
  return;
end initialize;

```

? but produce?

```

n = 8;          /* start table with 8 entries */
n_min = 8;     /* never let it get smaller than 8 */
lim1 = 0.33e0; /* contract table if only one-third full */
lim2 = 0.67e0; /* expand table when two-thirds full */
mult = 2;     /* double table when it is to be expanded */

```

```

call initialize;
return;

```

```

hash_funct: proc(key) fixed;
  /* this function will return a value between 1 and n, inclusive */
  dcl (key, t) fixed;
  t = mod(key, n-1) + 1;
  return t;
end hash_funct;

```

n-1
This comment is helpful?

```

dcl q ptr,
      failure label,
      (hash, i, j, old_n, new_n) fixed;

```

```

lookup: entry(newkey, newvalue, error);
dcl (newkey, newvalue) fixed, error label;
hash = hash_funct(newkey);
loop1:
do i = hash to n, 1 to hash-1;
  if p -> table(i).full then
do; if p -> table(i).skip then go to error; end;
  else
  if newkey = p -> table(i).key then
do;
  newvalue = p -> table(i).value;
  return;
end;
end loop1;
go to error; /* not found in full table - should be impossible */

```

```

enter: entry(newkey, newvalue, error);
hash = hash_funct(newkey);
loop2:
do i = hash to n, 1 to hash-1;
  if p -> table(i).full then
  do;
    p -> table(i).key = newkey;
    p -> table(i).value = newvalue;
    p -> table(i).full = "1"b;
    m = m + 1;
    if m > lim2*n then
    do;
      new_n = mult * n;
      failure = error;
      go to rehash;
    end;
  else return;
end;
end loop2;
go to error; /* table overflow - should be impossible */

```

{ dcl ok label, t fixed;
 ok = here; *← put label in*
 call lookup (newkey, t, ok); *here*
 go to error;
 here;

```

delete: entry(newkey, delerr);
dcl delerr label;
hash = hash_funct(newkey);
loop3:
do i = hash to n, 1 to hash-1;
  if p -> table(i).full then
  a3:
  do;
    if newkey = p -> table(i).key then
    b3:
    do;
      p -> table(i).full = "0"b;
      j = mod(i+1, n);
      p -> table(i).skip = /* mark skip */
      p -> table(j).full | p -> table(j).skip;
      m = m - 1;
      if m < lim1 * n then
      c3:
      do;
        new_n = n / mult;
        if new_n < n_min then return;
        failure = delerr;
        go to rehash;
      end c3;
      return;
    end b3;
  end a3;
  else
  if p -> table(i).skip then go to delerr; /* no entry found */
end loop3;
go to delerr; /* not found in full table - should be impossible */

```

rehash:

```
old_n = n;  
n = new_n;  
q = p;  
call initialize;  
loop4:  
do i = 1 to old_n;  
  if q -> table(i).full then  
    call enter(q -> table(i).key, q -> table(i).value, failure);  
  end loop4;  
  
free q -> table;  
return;  
  
end hash_table;
```

R 4.553+1.433