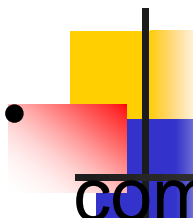
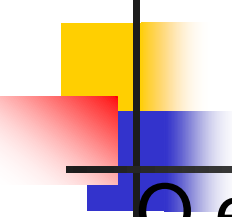


# Tabela *Hash*

- 
- Variações na técnica de pesquisa, conhecidas como *hashing*, têm sido implementadas em muitos compiladores.
  - *Hashing aberto* (“aberto” por não precisar haver limite no número de entradas que podem ser feitas numa tabela). Esse esquema nos dá ainda a capacidade de realizar  $e$  entradas sobre  $n$  nomes num tempo proporcional a  $n(n+e)/m$ , para qualquer constante  $m$  de nossa escolha.
  - Uma vez que  $m$  pode ser feita tão grande quanto desejamos, até o limite de  $n$ , esse método é geralmente mais eficiente do que as listas lineares. É o método de escolha para as tabelas de símbolos na maioria das situações [AHO, SETHI & ULLMAN, 86]

# Tabela *Hash*



O esquema básico de *hashing* é ilustrado na figura a seguir. Existem duas partes para a estrutura de dados [AHO, SETHI & ULLMAN, 86]:

1. Uma *tabela hash*, consistindo em um *array* fixo de  $m$  apontadores para entradas da tabela.
2. As entradas da tabela são organizadas em  $m$  listas ligadas separadas, chamadas de *buckets* (algumas das quais podem ficar vazias). Cada registro na tabela de símbolos figura exatamente em uma dessas listas. O armazenamento para os registros pode ser delineado a partir de um *array* de registros.

# Tabela *Hash*

Array de cabeçalhos de listas indexados pelo valor de hash

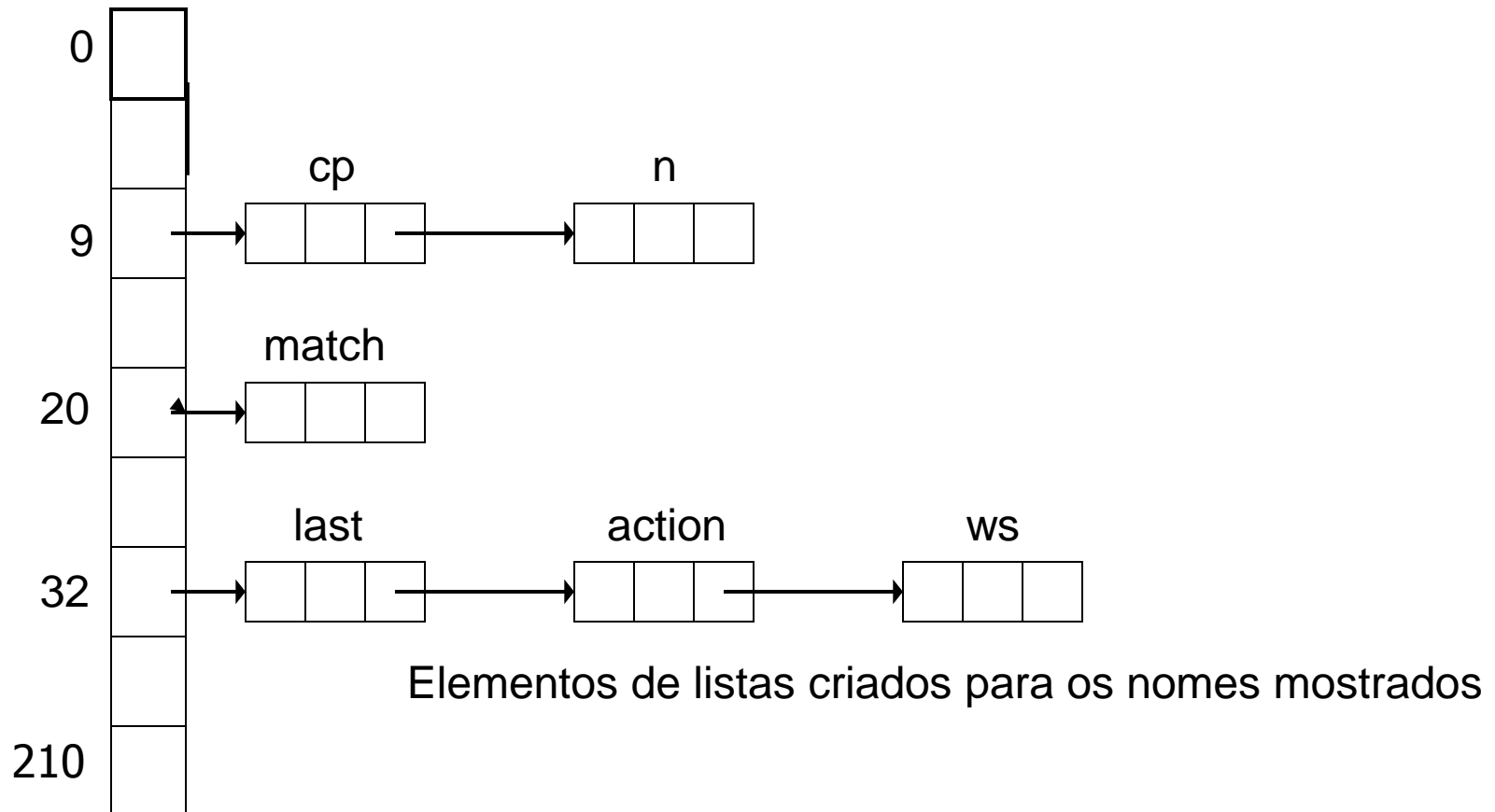
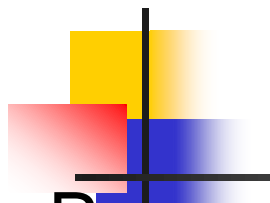
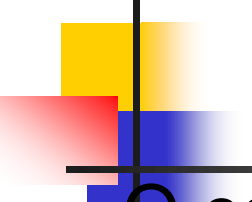


Figura: Uma tabela hash de tamanho 211.

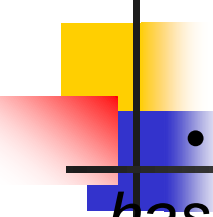
# Tabela *Hash*

- 
- Para se determinar se existe uma entrada para uma cadeia de caracteres  $s$  na tabela de símbolos, aplicamos uma *função de hash*  $h$  a  $s$  de tal forma que  $h(s)$  retorne um inteiro entre 0 e  $m-1$ .
  - Se  $s$  estiver na tabela de símbolos, estará na lista enumerada por  $h(s)$  e se ainda não estiver é introduzida através da criação de um registro para a mesma, que é ligado ao início da lista numerada por  $h(s)$ .
  - A lista média tem um comprimento de  $n/m$  registros, se existirem  $n$  nomes numa tabela de comprimento  $m$ .

# Tabela *Hash*

- 
- O espaço ocupado pela tabela de símbolos consiste de  $m$  palavras para a tabela *hash* e  $cn$  palavras para as entradas da tabela, onde  $c$  é o número de palavras de entrada na tabela. Por conseguinte, o espaço para a tabela *hash* depende somente de  $m$  e o espaço para as entradas da tabela depende somente do número de entradas.
  - Uma boa atenção tem sido dada à questão de como projetar uma função de *hash* que seja fácil de computar para as cadeias de caracteres, além de distribuí-las uniformemente dentre as  $m$  listas.

# Tabela *Hash*



- Uma abordagem adequada ao cômputo das funções de *hash* está em se proceder da seguinte forma [AHO, SETHI & ULLMAN, 86]:

1. determinar um inteiro positivo  $h$  a partir dos caracteres  $c_1, c_2, \dots, c_k$  na cadeia  $s$ . A conversão de caracteres singelos para inteiros é usualmente suportada pela linguagem de programação. Pascal providencia a função *ord* com esse propósito; C converte automaticamente um caracter para um inteiro, se uma operação aritmética for realizada sobre o mesmo;
2. converter o inteiro  $h$  determinado acima no número de uma lista, isto é, um inteiro entre 0 e  $m-1$ . Simplesmente dividir por  $m$  e ficar com o resto é uma política razoável. Ficar com resto parece funcionar melhor se  $m$  for primo (ver na figura  $m=211$ ).

# Tabela *Hash*

- Uma técnica simples para computar  $h$  está em adicionar os valores inteiros dos caracteres na cadeia. Uma idéia melhor está em multiplicar o valor antigo de  $h$  por uma constante  $\alpha$  antes de adicionar o próximo caractere. Isto é, fazendo-se  $h_0=0$ ,  $h_i = \alpha h_{i-1} + c_j$ , para  $1 \leq i \leq k$  e seja  $h = h_k$ , onde  $k$  é o tamanho da cadeia (relembremos que o valor de *hash* que dá o número de listas é  $h \bmod m$ ).