

## Continuação do Programa E.21

```

while (!EExterno(p))
{
    if (Bit(p->NO.NInterno.Index, k) == 1)
        p = p->NO.NInterno.Dir;
    else p = p->NO.NInterno.Esq;
}
/* acha o primeiro bit diferente */
i = 1;
while (((i <= D) & (Bit((int)i, k) == Bit((int)i, p->NO.Chave)))
        i++);
if (i > D)
{
    printf("Erro: chave ja esta na arvore\n"); return (*t);
}
else return (InsereEntre(k, t, i));
}
}

```

## Programa E.22 Geração de pesos para a função de transformação

```

void GeraPesos(TipoPesos p)
{
    int i;
    struct timeval semente;
    /* Utilizar o tempo como semente para a funcao srand() */
    gettimeofday(&semente, NULL);
    srand((int)(semente.tv_sec + 1000000*semente.tv_usec));
    for (i = 0; i < n; i++)
        p[i] = 1+(int)(10000.0*rand()/(RAND_MAX+1.0));
}

```

## Programa E.23 Implementação de função de transformação

```

typedef char TipoChave[N];

TipoIndice h(TipoChave Chave, TipoPesos p)
{
    int i;
    unsigned int Soma = 0;
    int comp = strlen(Chave);
    for (i = 0; i < comp; i++) Soma += (unsigned int)Chave[i] * p[i];
    return (Soma % M);
}

```

## Programa E.24 Geração de pesos para a função de transformação hz

```

#define TAMALFABETO 256
typedef unsigned TipoPesos[N][TAMALFABETO];

void GeraPesos(TipoPesos p)
{
    /* Gera valores randomicos entre 1 e 10.000 */
}

```

## Continuação do Programa E.24

```

int i, j;
struct timeval semente;
/* Utilizar o tempo como semente para a funcao srand() */
gettimeofday(&semente, NULL);
srand((int)(semente.tv_sec + 1000000 * semente.tv_usec));
for (i = 0; i < N; i++)
    for (j = 0; j < TAMALFABETO; j++)
        p[i][j] = 1 + (int)(10000.0 * rand() / (RAND_MAX + 1.0));
}

```

## Programa E.25 Implementação de função de transformação hz

```

typedef char TipoChave[N];

TipoIndice h(TipoChave Chave, TipoPesos p)
{
    int i; unsigned int Soma = 0;
    int comp = strlen(Chave);
    for (i = 0; i < comp; i++) Soma += p[i][(unsigned int)Chave[i]];
    return (Soma % M);
}

```

## Programa E.26 Estrutura do dicionário para listas encadeadas

```

typedef char TipoChave[N];
typedef unsigned TipoPesos[N][TAMALFABETO];
typedef struct Tipoltem {
    /* outros componentes */
    TipoChave Chave;
} Tipoltem;

typedef unsigned int TipoIndice;
typedef struct TipoCelula* TipoApontador;
typedef struct TipoCelula {
    Tipoltem Item;
    TipoApontador Prox;
} TipoCelula;

typedef struct TipoLista {
    TipoCelula *Primeiro, *Ultimo;
} TipoLista;

typedef TipoLista TipoDicionario[M];

```

## Programa E.27 Operações do Dicionário usando listas encadeadas

```

void Inicializa (TipoDicionario T)
{
    int i;
    for (i = 0; i < M; i++) FLVazia(&T[i]);
}

```