Programa E.36 *Teste para a função de transformação perfeita*

```
#define MAXNUMVERTICES 100000 /*--No. maximo de vertices--*/
#define MAXNUMARESTAS 100000 /*--No. maximo de arestas--*/
#define MAXR 5
#define MAXTAMCHAVE 6 /*--No. maximo de caracteres da chave--*/
#define MAXNUMCHAVES 100000 /*--No. maximo de chaves lidas--*/
typedef int TipoValorVertice;
typedef int TipoValorAresta;
typedef int Tipor;
typedef int TipoPesos[MAXTAMCHAVE];
typedef TipoPesos TipoTodosPesos[MAXR];
typedef int Tipog[MAXNUMVERTICES];
typedef char TipoChave[MAXTAMCHAVE];
typedef TipoChave TipoConjChaves[MAXNUMCHAVES];
typedef TipoValorVertice TipoIndice;
static TipoValorVertice M;
static TipoValorAresta N;

/*-- Entra aqui a funcao hash universal do Programa E.23 --*/
/*-- Entra aqui a funcao hash perfeita do Programa E.35 --*/
int main()
{ Tipor r;
  Tipog g;
  TipoTodosPesos Pesos;
  int i, j;
  TipoConjChaves ConjChaves;
  FILE *ArqChaves;
  FILE *ArqFHPM;
  char NomeArq[100];
  TipoChave Chave;
  inline short VerificaFHPM()
  { short TabelaHash[MAXNUMVERTICES];
    int   i, indiceFHPM;
    for (i = 0; i < N; i++) TabelaHash[i] = FALSE;
    for (i = 0; i < N; i++)
      { indiceFHPM = hp (ConjChaves[i], r, Pesos, g);
        if ((TabelaHash[indiceFHPM])||(indiceFHPM >= N)) return FALSE;
        TabelaHash[indiceFHPM] = TRUE;
      }
    return TRUE;
  }
  printf ("Nome do arquivo com chaves a serem lidas: ");
  scanf("%s*[^\n]", NomeArq);
  printf("NomeArq = %s\n", NomeArq);
  ArqChaves = fopen(NomeArq, "r");
  fscanf(ArqChaves, "%d %d %d*[^\n]", &N, &M, &r);
  Ignore(ArqChaves, '\n');
  printf("N=%d, M=%d, r=%d\n", N, M, r);
  i = 0;
```


Continuação do Programa E.36

```
  while ((i < N) && (!feof(ArqChaves)))
  { fscanf(ArqChaves,"%s*[^\n]", ConjChaves[i]);
    Ignore(ArqChaves,'\n');
    printf("Chave[%d]=%s\n", i, ConjChaves[i]);
    i++;
  }
  if (i != N)
  { printf("Erro: entrada com menos do que ', N, ' elementos.\n");
    exit(-1);
  }
  printf ("Nome do arquivo com a funcao hash perfeita: ");
  scanf("%s*[^\n]", NomeArq);
  printf("NomeArq = %s\n", NomeArq);
  ArqFHPM = fopen(NomeArq, "rb");
  fscanf(ArqFHPM, "%d*[^\n]", &N); Ignore(ArqFHPM, '\n');
  fscanf(ArqFHPM, "%d*[^\n]", &M); Ignore(ArqFHPM, '\n');
  fscanf(ArqFHPM, "%d*[^\n]", &r); Ignore(ArqFHPM, '\n');
  printf("N=%d, M=%d, r=%d\n", N, M, r);
  for (j = 0; j < r; j++)
  { for (i = 0; i < MAXTAMCHAVE; i++)
      fscanf(ArqFHPM, "%d*[^%d\n]", &Pesos[j][i]);
    Ignore(ArqFHPM, '\n');
    printf("\n");
    for (i = 0; i < MAXTAMCHAVE; i++)
      printf("%d ", Pesos[j][i]);
    printf("  (p%d)\n", j);
  }
  for (i = 0; i < M; i++)
    fscanf(ArqFHPM, "%d*[%d\n]", &g[i]);
  Ignore(ArqFHPM, '\n');
  for (i = 0; i < M; i++) printf("%d ", g[i]);
  printf("  (g)\n");
  if (VerificaFHPM())
  printf ("FHPM foi gerada com sucesso\n");
  else printf ("FHPM nao foi gerada corretamente\n");
  printf("Chave: ");
  scanf("%s*[^\n]", Chave);
  while (strcmp(Chave, "aaaaaa") != 0)
  { printf ("FHPM: %d\n", hp(Chave, r, Pesos, g));
    printf("Chave: ");
    scanf("%s*[^\n]", Chave);
  }
  fclose (ArqChaves);
  fclose (ArqFHPM);
  return 0;
}
```