```c
// Hash_main_char.c
// ハッシュ法の動作検証用 main 関数

#include "Hash_main_char.h" // ハッシュ法 main 関数用定義ファイル
#define HSIZE 100000        // ハッシュ表の大きさ(集合の濃度: 10 万)
#include "Measurement_1a.h" // 回数測定用定義ファイル

// パラメータ設定
// 除数(DVS),ハッシュ増分値(CONSL)の値は学籍番号末尾桁の値によって異なる
#define DVS 99991
#define CONSL  19

#include "Hash_open_char.h" // 開放番地法

int main(void) {
  FILE *fin;            // 入力用ファイルポインタ
  int i,                // for ループ用変数
      rc,               // 関数の戻り値
      itemNo = 0,       // 処理要素数
      oun = HSIZE / 10; // 処理レコード数出力単位
  char inv[LSTR];       // 入力値(集合の要素)

  fin = open_file(FILENAME, "r");  // 入力ファイル(整数集合)をオープン
  Htable = allocate_Htable(HSIZE); // ハッシュ表の動的割付
  init_tbl(Htable, HSIZE);         // ハッシュ表を初期化

  printf("DVS: %d CONSL: %d¥n¥n", DVS, CONSL); // パラメータの出力
  printf("要素数 衝突番地総数 衝突キー総数 最大同族数 最小同族数 平均同族数¥n");

  while (fscanf(fin, FMTF, inv) != EOF) {  // データファイルから入力値を読み込む
    STORE;                                 // 要素をハッシュ表に格納
    if (++itemNo % oun == 0) {             // 1U レコードごとに計測値を出力
      for (i = 0; i < HSIZE; i++)          // ハッシュ表を先頭番地から照査
        if (TABLE[i] > 1) {                // i 番地は衝突が発生している
          clsnAdrs++;                      // 衝突番地数を加算
          clsnKeys += TABLE[i];            // 衝突キー数を加算
          if (maxSynm < TABLE[i])          // 現在の最大同族数 < 当該番地の同族数
            maxSynm = TABLE[i];            // 最大同族数を更新
          if (minSynm == 0)                // 現在の最小同族数は初期値
            minSynm = TABLE[i];            // 最小同族数を更新
          else if (minSynm > TABLE[i])     // 現在の最小同族数 > 当該番地の同族数
            minSynm = TABLE[i];            // 最小同族数を更新
        }
      aveSynm = (clsnKeys == 0) ? 0 : (clsnKeys / clsnAdrs); // 平均同族数の算出
      printf("%5dU %12d %12d %10d %10d %10d¥n",
             itemNo/oun, clsnAdrs, clsnKeys, maxSynm, minSynm, aveSynm);
      clsnAdrs = 0;
      clsnKeys = 0;
    }
  }

  fclose(fin); // 開いたファイルをクローズ
  return 0;    // 終了
}
```