Xử lý đụng độ:

* Dò tuyến tính

int HashModM(int k, int M)

{

if (M == 0)

return 0;

return (k % M);

}

// Hàm tìm vị trí trống dùng phương pháp dò tuyến tính

int LinearProbing(int hashValue, int i, int M)

{

return (hashValue + i) % M;

}

// Hàm để thêm sinh viên vào bảng băm

void InsertStudent(int studentID, string name)

{

int hashValue = HashModM(studentID, TABLE\_SIZE);

int index = hashValue;

int iHashingTime = 0;

// Dò tuyến tính để tìm vị trí trống

while (hashTable[index].studentID != 0)

{

iHashingTime++; // hashing lan thu i

index = LinearProbing(hashValue, iHashingTime, TABLE\_SIZE);

// Nếu quay lại vị trí ban đầu thì dừng (không có vị trí trống)

if (index == hashValue)

return;

}

// Lưu sinh viên vào vị trí đã tìm được

hashTable[index].studentID = studentID;

hashTable[index].name = name;

cout << "Inserted student " << name << " with ID " << studentID << " at index " << index << endl;

}

* **Băm kép:**

// Hàm băm chính (dùng phép chia lấy dư)

int Hash1(int k, int M)

{

if (M == 0)

return 0;

return k % M;

}

int Hash2(int k, int M)

{

if (M < 2)

return 0;

return (k % (M - 1)) + 1;

}

int Hash3(int k, int M)

{

if (M < 3)

return 0;

return (M - 2) - k % (M - 2);

}

// Hàm băm kép

int DoubleHash(int k, int i, int M)

{

return (Hash1(k, M) + i \* Hash2(k, M)) % M;

}

int DoubleHash2(int k, int i, int M)

{

return (Hash1(k, M) + i \* Hash3(k, M)) % M;

}