Tên: Vũ Tuấn Hải

MSSV: 17520433

Lớp: IT007.J21

**BÁO CÁO LAB 6**

**6.4. HƯỚNG DẪN THỰC HÀNH**

Mô phỏng 3 giải thuật thay thế trang: FIFO, OPT, LRU

a. Source code

#include <iostream>

using namespace std;

static int preArray[10];

static int page\_fault[100];

static int number\_page\_fault = 1;

int Is\_in\_preArray(int page, int value) {

for (int i = 0; i < page; i++) {

if (value == preArray[i]) return i;

}

return -1;

}

int Farest\_Element(int \*ref, int pivot, int page) {

int min = pivot;

int vt;

for (int i = 0; i < page; i++) {

for (int j = pivot - 1; j >= 0; j--) {

if (preArray[i] == ref[j]) {

if (j < min) {

min = j;

vt = i;

}

break;

}

}

}

return vt;

}

int Farest\_Element\_Oppsite(int \*ref, int pivot, int page, int n) {

int max = pivot;

int flag[10];

for (int i = 0; i < page; i++) {

flag[i] = false;

}

int vt = -1;

for (int i = 0; i < page; i++) {

for (int j = pivot + 1; j < n; j++) {

if (preArray[i] == ref[j]) {

if (j > max) {

max = j;

vt = i;

}

flag[i] = true;

break;

}

}

}

for (int i = 0; i < page; i++) {

if (flag[i] == false) return i;

}

return vt;

}

void Print(int total\_page[10][100], int n, int page, int ref[100]) {

// Print

for (int i = 0; i < n; i++) {

cout << ref[i] << " ";

}

cout << endl;

for (int i = 0; i < page; i++) {

for (int j = 0; j < n; j++) {

if (total\_page[i][j] != -1) {

cout << total\_page[i][j] << " ";

}

else {

cout << " ";

}

}

cout << endl;

}

for (int i = 0; i < n; i++) {

if (page\_fault[i] == 1) cout << "\* ";

else {

cout << " ";

}

}

cout << endl;

cout << "Number of Page Fault: " << number\_page\_fault << endl;

}

void FIFO(int ref[], int n, int page)

{

bool IsFault;

int total\_page[10][100];

int current\_page = 0;

for (int i = 0; i < page; i++) {

if (i == 0) { total\_page[i][0] = ref[0]; }

else {

total\_page[i][0] = -1;

}

}

page\_fault[0] = 1;

for (int j = 1; j < n; j++) {

for (int i = 0; i < page; i++) {

total\_page[i][j] = total\_page[i][j-1];

preArray[i] = total\_page[i][j - 1];

}

if (Is\_in\_preArray(page, ref[j]) != -1) {

page\_fault[j] = -1;

}

else {

current\_page++;

if (current\_page == page) current\_page = 0;

total\_page[current\_page][j] = ref[j];

page\_fault[j] = 1;

number\_page\_fault++;

}

}

Print(total\_page, n, page, ref);

}

void OPT(int ref[], int n, int page)

{

bool IsFault;

int total\_page[10][100];

int current\_page = 0;

for (int i = 0; i < page; i++) {

if (i == 0) { total\_page[i][0] = ref[0]; }

else {

total\_page[i][0] = -1;

}

}

page\_fault[0] = 1;

for (int j = 1; j < n; j++)

{

for (int i = 0; i < page; i++) {

total\_page[i][j] = total\_page[i][j - 1];

preArray[i] = total\_page[i][j - 1];

}

///////

if (Is\_in\_preArray(page, ref[j]) != -1) {

page\_fault[j] = 0;

}

else {

if (j < page) {

current\_page++;

total\_page[current\_page][j] = ref[j];

}

else {

int pivot = Farest\_Element(ref, j, page);

total\_page[pivot][j] = ref[j];

}

page\_fault[j] = 1;

number\_page\_fault++;

}

/////

}

Print(total\_page, n, page, ref);

}

void LRU(int ref[], int n, int page)

{

bool IsFault;

int total\_page[10][100];

int current\_page = 0;

for (int i = 0; i < page; i++) {

if (i == 0) { total\_page[i][0] = ref[0]; }

else {

total\_page[i][0] = 0;

}

}

page\_fault[0] = 1;

for (int j = 1; j < n; j++)

{

for (int i = 0; i < page; i++) {

total\_page[i][j] = total\_page[i][j - 1];

preArray[i] = total\_page[i][j - 1];

}

///////

if (Is\_in\_preArray(page, ref[j]) != -1) {

page\_fault[j] = 0;

}

else {

if (j < page) {

current\_page++;

total\_page[current\_page][j] = ref[j];

}

else {

int pivot = Farest\_Element\_Oppsite(ref, j, page, n);

total\_page[pivot][j] = ref[j];

}

page\_fault[j] = 1;

number\_page\_fault++;

}

/////

}

Print(total\_page, n, page, ref);

}

int main() {

int page; int temp;

int ref[11] = { 1, 7, 5, 2, 0, 4, 3, 3, 0, 0, 7 };;

int n = 11;

cout << "--- Page Replacement algorithm ---" << endl;

cout << "1. Default referenced sequence" << endl;

cout << "2. Manual input sequence" << endl;

cin >> temp;

switch (temp) {

case 1:

break;

case 2:

cout << "Nhap so luong: "; cin >> n;

cout << "Nhap danh sach trang: ";

for (int i = 0; i < n; i++) {

cin >> ref[i];

}

}

cout << "--- Page Replacement algorithm ---" << endl;

cout << "Input page frames: "; cin >> page;

cout << "--- Select algorithm ---" << endl;

cout << "1. FIFO algorithm" << endl;

cout << "2. OPT algorithm" << endl;

cout << "3. LRU algorithm" << endl;

cout << "--- Enter input ---" << endl;

cin >> temp;

cout << "--- Page Replacement algorithm--- " << endl;

switch (temp) {

case 1:

FIFO(ref, n, page);

break;

case 2:

OPT(ref, n, page);

break;

case 3:

LRU(ref, n, page);

break;

}

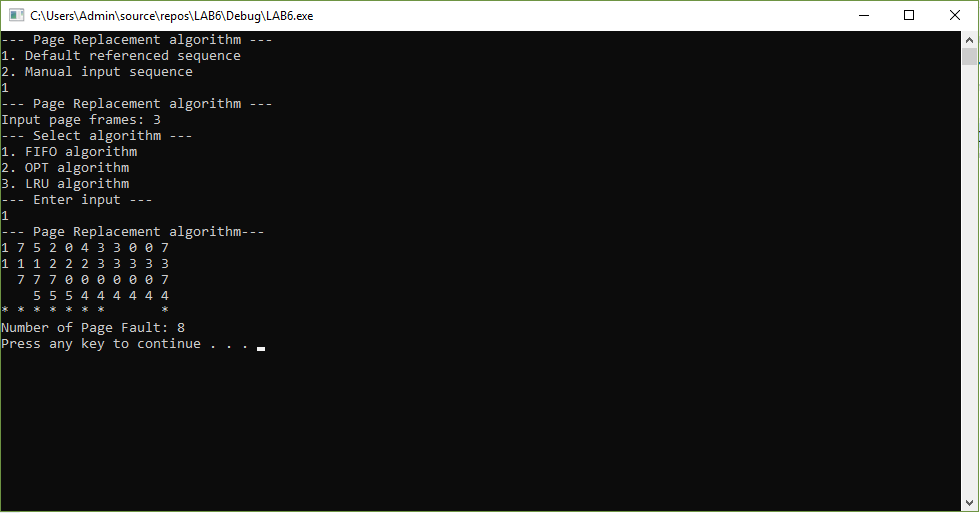
system("pause");

return 0;

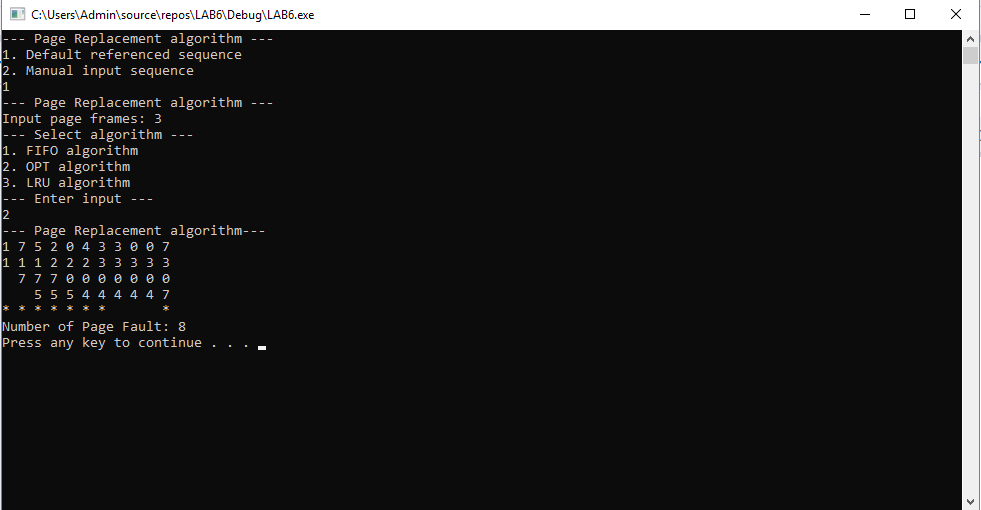
}

1. Kết quả chạy thử

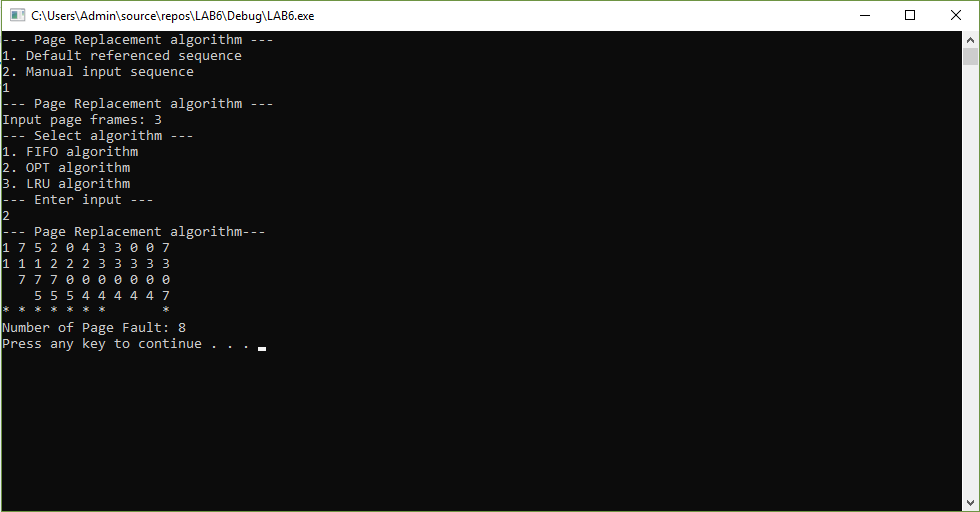
* FIFO



* OPT



* LRU

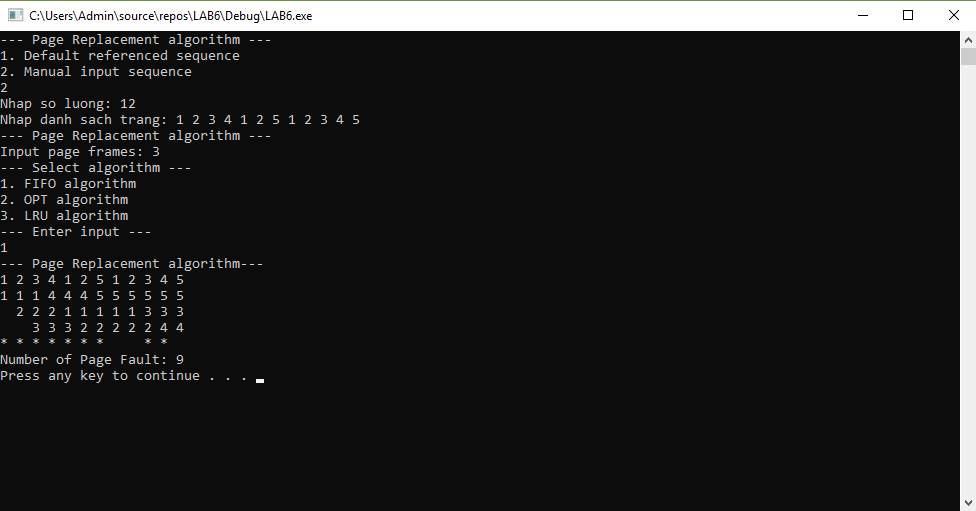


**6.5. BÀI TẬP ÔN TẬP**

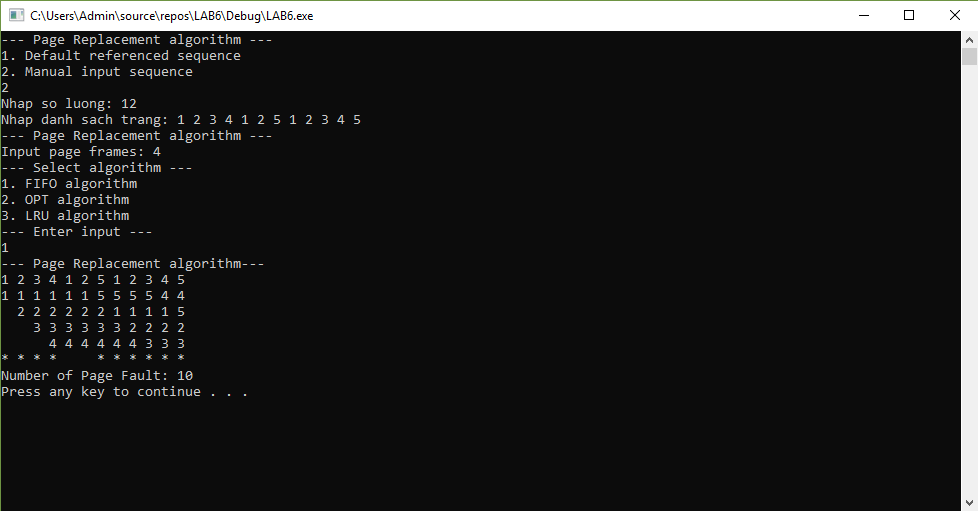
Nghịch lý Belady là hiện tượng số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng.

Chứng minh: ví dụ: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Sử dụng 3 khung trang , sẽ có 9 lỗi trang.



Sử dụng 4 khung trang, sẽ có 10 lỗi trang



2.

Giải thuật FIFO: dễ cài đặt, dễ hiện thực, hiệu quả kém

Giải thuật LRU: khó cài đặt, phức tạp, hiệu quả

Giải thuật OPT: không khả thi, nhưng hiệu quả nhất

* Giải thuật OPT bất khi thi nhất vì hoàn toàn không biết được thứ tự các trang trong tương lai.
* Giải thuật OPT và LRU là phức tạp nhất bởi vì mỗi lần lỗi trang, khi tìm khung trang thích hợp để thay thế thì phải xét đến toàn bộ chuỗi tham chiếu trước / sau nó.