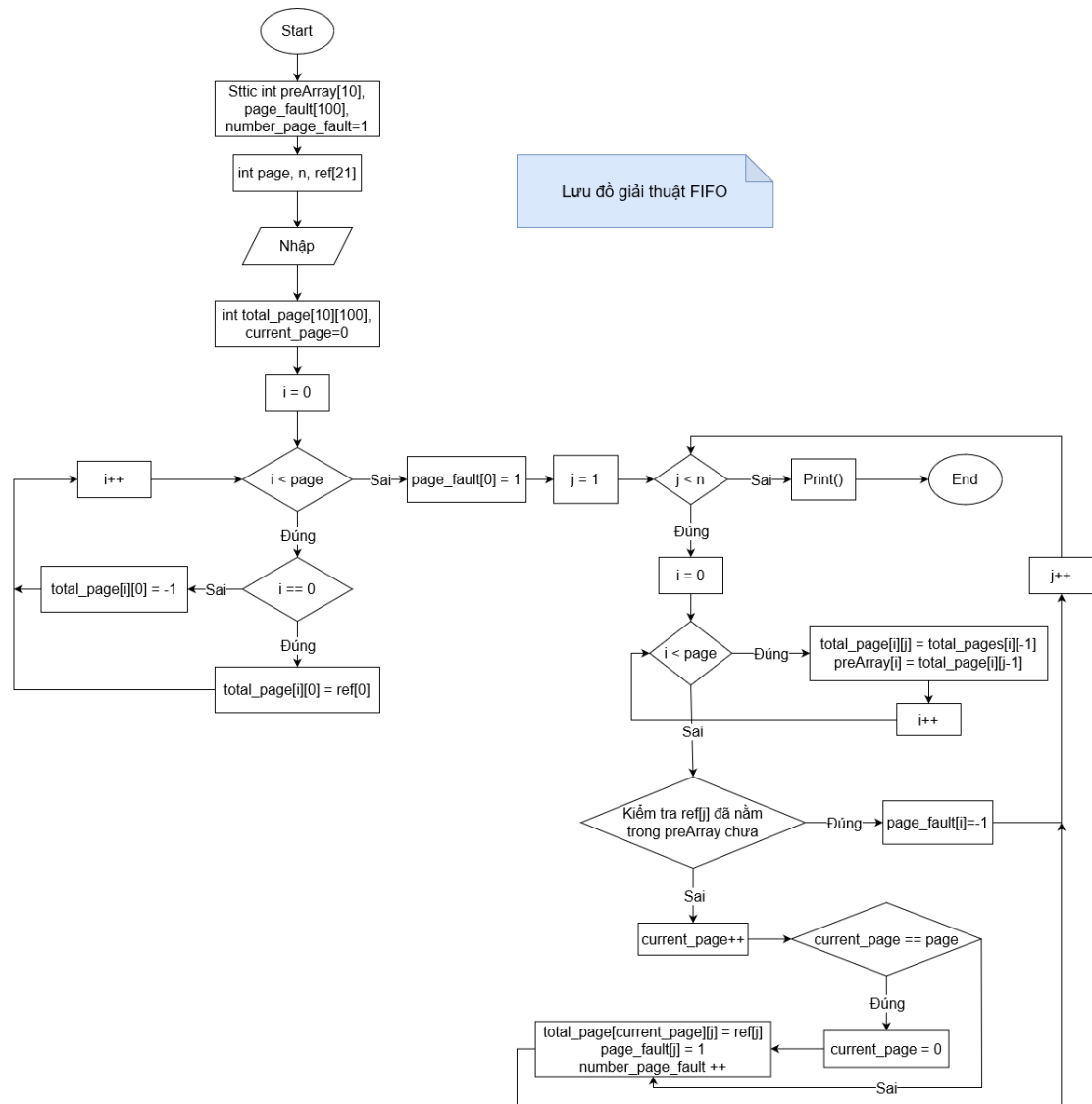


## Section 6.4

### 1. Task name 1: Giải thuật FIFO

#### 🚦 Lưu đồ thuật toán



Hình 1: Lưu đồ thuật toán FIFO

#### 🚦 Giải thích

- Bước 1: Khai báo các biến cần thiết như là: `ref[100]` là mảng các trang, `n` là số trang (cùng là chiều dài của mảng `ref`), `page` là số khung trang. Sau đó tiến hành xây dựng hàm nhập để nhập các thông số vào 3 biến này. Ngoài ra còn cần các biến như 1 mảng chứa các thông tin của một cột, 1 mảng để lưu các trang bị lỗi và 1 biến đếm các trang bị lỗi.

- Bước 2: Tạo mảng 2 chiều `total_page[][]` là cái bảng lưu trữ thông tin, là biến `curent_page` để tác định trang mình đang làm việc.
- Bước 3: Điền chữ số đầu tiên của mảng `ref` vào cột đầu tiên của mảng 2 chiều, các trang còn lại của cột mà ko có giá trị sẽ được điền bằng -1.
- Bước 4: Tiến hành xét các giá trị tiếp theo của mảng.
- Bước 5: Sao chép cột đang xét bằng cột ở trước đó, Đồng thời sao chép là mảng `preArray[]`, cũng là giá trị của cột trước đó để tiến hành xét với giá trị trong mảng là `ref[j]` đang xét. Nếu giá trị `ref[j]` nằm ở trong mảng rồi thì tiến hành `j++` để thực hiện các tiến trình tiếp theo. Nếu sai thì xét từng giá trị của `page` để tìm ra vị trí cần thay thế và đánh dấu lỗi trang lúc, tiến hành xét các giá trị tiếp theo cho đến hết.
- Bước 6: Sau khi lặp hết các giá trị trong mảng `ref[]` thì tiến hành in bảng `total_page` và các giá trị lỗi sai.

### Test Case

- Ví dụ 1: Xét chuỗi truy xuất bộ nhớ sau: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế FIFO, giả sử có 5 khung trang?

+ Lời giải:

Giải bằng tay:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2
			4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
*	*	*	*			*	*		*	*	*	*							

=> Tổng cộng có 10 lỗi trang.

Giải bằng code:

```

the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 20
Nhap danh sach trang: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
--- Page Replacement algorithm ---
Input page frames: 5
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6
2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3
5 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7
* * * * *
Number of Page Fault: 10

```

Hình 2: Kết quả giải ví dụ 1 bằng thuật toán FIFO

- Ví dụ 2: Xét chuỗi truy xuất bộ nhớ sau: 1, 3, 4, 5, 2, 3, 6, 3. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế FIFO, giả sử có 4 khung trang?

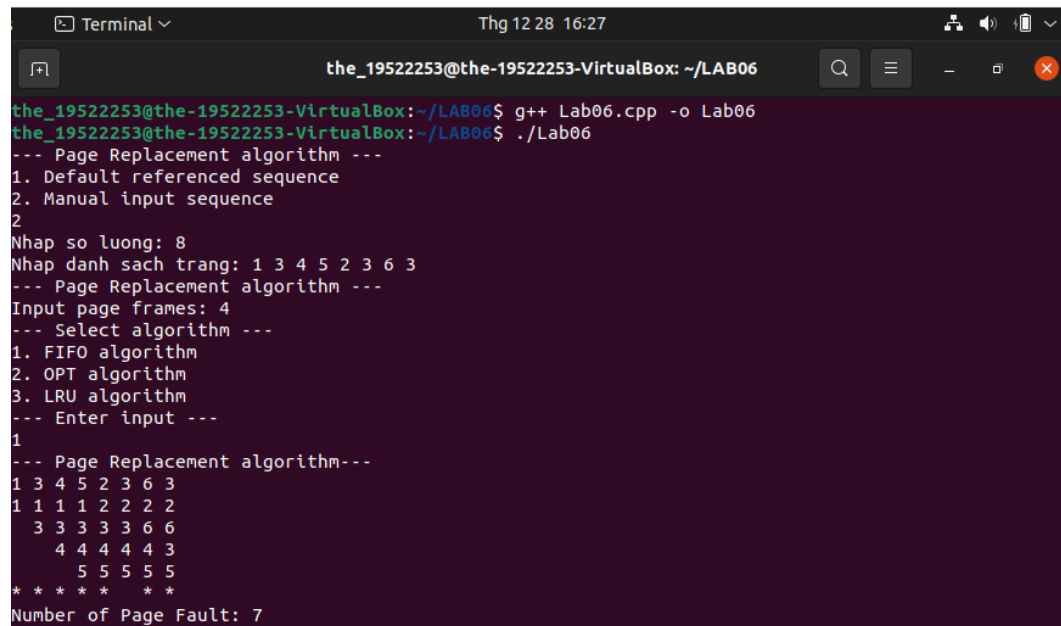
+ Lời giải:

Giải bằng tay:

1	3	4	5	2	3	6	3
1	1	1	1	2	2	2	2
	3	3	3	3	3	6	6
		4	4	4	4	4	3
			5	5	5	5	5
*	*	*	*	*		*	*

=> Tổng cộng có 7 lỗi trang.

Giải bằng code:



```

the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 8
Nhập danh sách trang: 1 3 4 5 2 3 6 3
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
1 3 4 5 2 3 6 3
1 1 1 1 2 2 2 2
  3 3 3 3 3 6 6
    4 4 4 4 4 3
      5 5 5 5 5
* * * * *
Number of Page Fault: 7
  
```

Hình 3: Kết quả giải ví dụ 2 bằng thuật toán FIFO

- Ví dụ 3: Xét chuỗi truy xuất bộ nhớ sau: 6, 2, 4, 4, 5, 6, 3, 1, 4, 2, 3, 7, 5, 6, 7, 2, 4, 3, 5, 1. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế FIFO, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

6	2	4	4	5	6	3	1	4	2	3	7	5	6	7	2	4	3	5	1
6	6	6	6	6	6	3	3	3	3	3	3	5	5	5	5	5	5	5	1
	2	2	2	2	2	2	1	1	1	1	1	1	6	6	6	6	6	6	6
		4	4	4	4	4	4	4	2	2	2	2	2	2	2	4	4	4	4
				5	5	5	5	5	5	5	7	7	7	7	7	7	3	3	3
*	*	*		*		*	*		*		*	*	*			*	*		*

=> Tổng cộng có 13 lỗi trang.

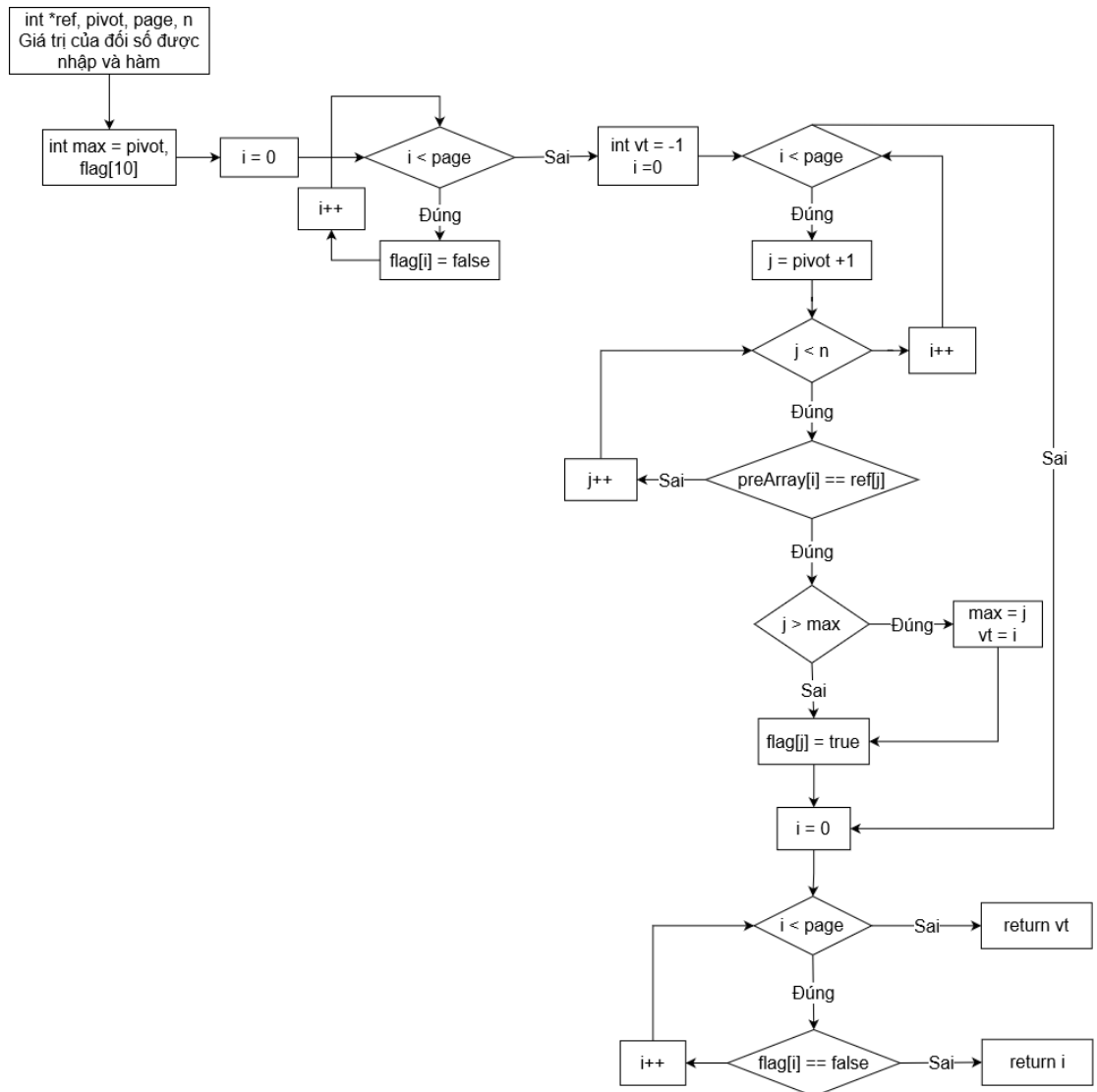
Giải bằng code:

```
the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 20
Nhap danh sach trang: 6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1
6 6 6 6 6 6 3 3 3 3 3 3 5 5 5 5 5 5 5 1
  2 2 2 2 2 2 1 1 1 1 1 1 6 6 6 6 6 6 6 6
    4 4 4 4 4 4 4 2 2 2 2 2 2 2 4 4 4 4
      5 5 5 5 5 5 5 7 7 7 7 7 7 7 3 3 3 3
* * * * *
Number of Page Fault: 13
```

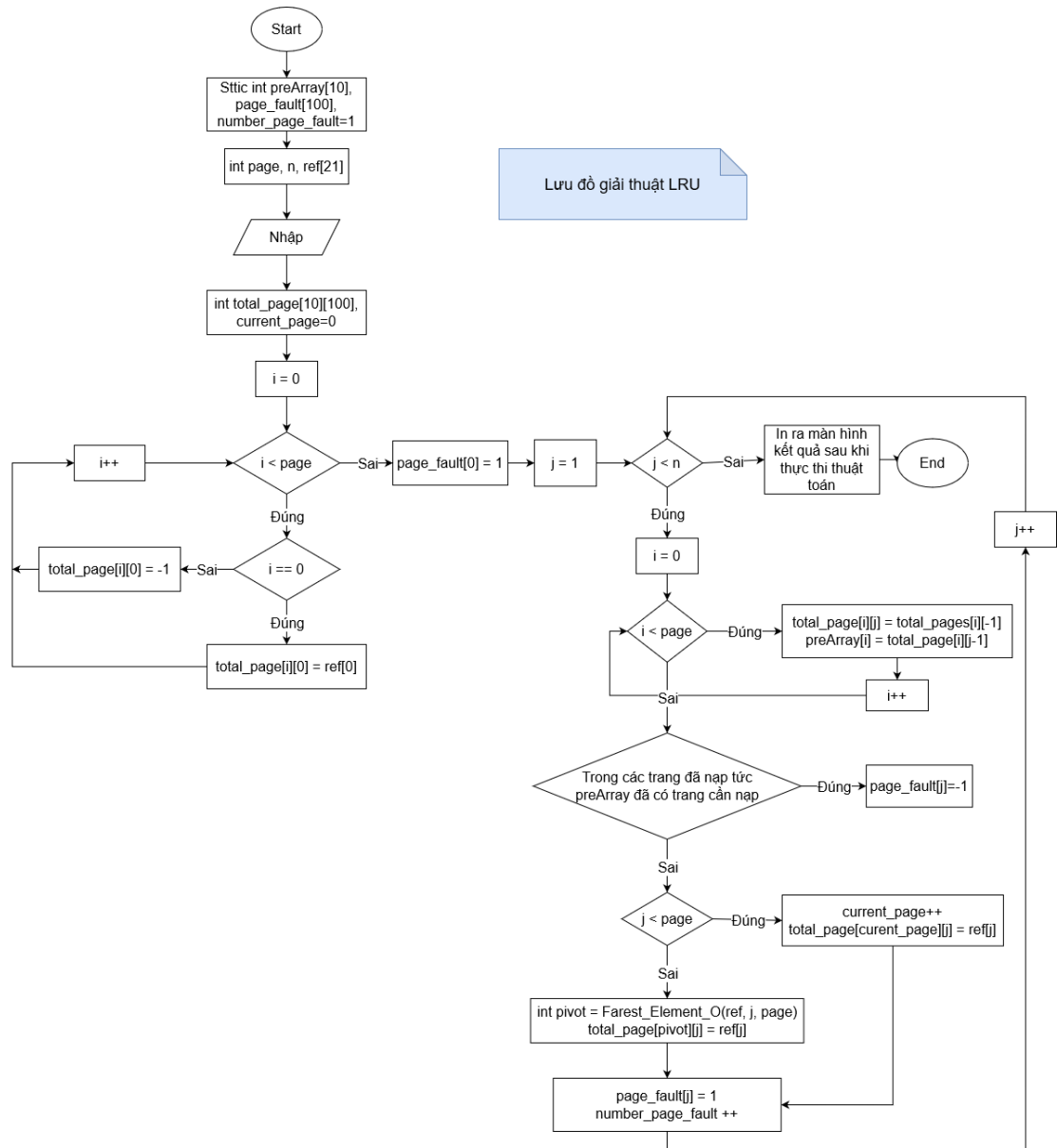
Hình 4: Kết quả giải ví dụ 3 bằng thuật toán FIFO

## 2. Task name 2: Giải thuật LRU

### Lưu đồ thuật toán



Hình 5: Lưu đồ hàm tìm ra trang được gói sớm nhất trong quá khứ



Hình 6: Lưu đồ thuật toán LRU.

### Giải thích

- Hình 5: Tiến hành lọc quá tất cả các trang có trong cột và từ đó xét với mảng ref để xem trang nào được truy xuất sớm nhất trong quá khứ.
- Hình 6: Cũng tương tự như thuật toán OPT thì ở Bước 5, để chọn ra vị trí page cần thay thế thì ta sẽ sử dụng hàm **Farest\_Element\_O** để xác định vị trí của trang được truy xuất sớm nhất trong quá khứ. Còn lại tất cả các bước thì như nhau.

## Test Case

- Ví dụ 1: Xét chuỗi truy xuất bộ nhớ sau: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế LRU, giả sử có 5 khung trang?

+ Lời giải:

Giải bằng tay:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
*	*	*	*			*	*				*	*							

=> Tổng cộng có 8 lỗi trang.

Giải bằng code:

```

the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 20
Nhập danh sách trang: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
--- Page Replacement algorithm ---
Input page frames: 5
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- Page Replacement algorithm---
1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
0 0 0 4 4 4 5 6 6 6 6 6 6 6 6 6 6 6 6 6
0 0 0 0 0 0 0 0 0 0 0 0 7 7 7 7 7 7 7 7
* * * * *
Number of Page Fault: 7

```

Hình 7: Kết quả giải ví dụ 1 bằng thuật toán LRU



- Ví dụ 2: Xét chuỗi truy xuất bộ nhớ sau: 1, 3, 4, 5, 2, 3, 6, 3. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế LRU, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

1	3	4	5	2	3	6	3
1	1	1	1	2	2	6	6
	3	3	3	3	3	3	3
		4	4	4	4	4	4
			5	5	5	5	5
*	*	*	*	*		*	

=> Tổng cộng có 6 lỗi trang.

Giải bằng code:

```

the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 8
Nhập danh sách trang: 1 3 4 5 2 3 6 3
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- Page Replacement algorithm---
1 3 4 5 2 3 6 3
1 1 1 1 2 2 6 6
0 3 3 3 3 3 3 3
0 0 4 4 4 4 4 4
0 0 0 5 5 5 5 5
* * * * *
Number of Page Fault: 6

```

Hình 8: Kết quả giải ví dụ 2 bằng thuật toán LRU

- Ví dụ 3: Xét chuỗi truy xuất bộ nhớ sau: 6, 2, 4, 4, 5, 6, 3, 1, 4, 2, 3, 7, 5, 6, 7, 2, 4, 3, 5, 1. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế LRU, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

6	2	4	4	5	6	3	1	4	2	3	7	5	6	7	2	4	3	5	1
6	6	6	6	6	6	6	6	6	2	2	2	2	6	6	6	6	3	3	3
	2	2	2	2	2	3	3	3	3	3	3	3	3	3	2	2	2	2	1
		4	4	4	4	4	1	1	1	1	7	7	7	7	7	7	7	5	5
				5	5	5	5	4	4	4	4	5	5	5	5	4	4	4	4
*	*	*		*		*	*	*	*		*	*	*		*	*	*	*	*

=> Tổng cộng có 16 lỗi trang.

Giải bằng code:

```

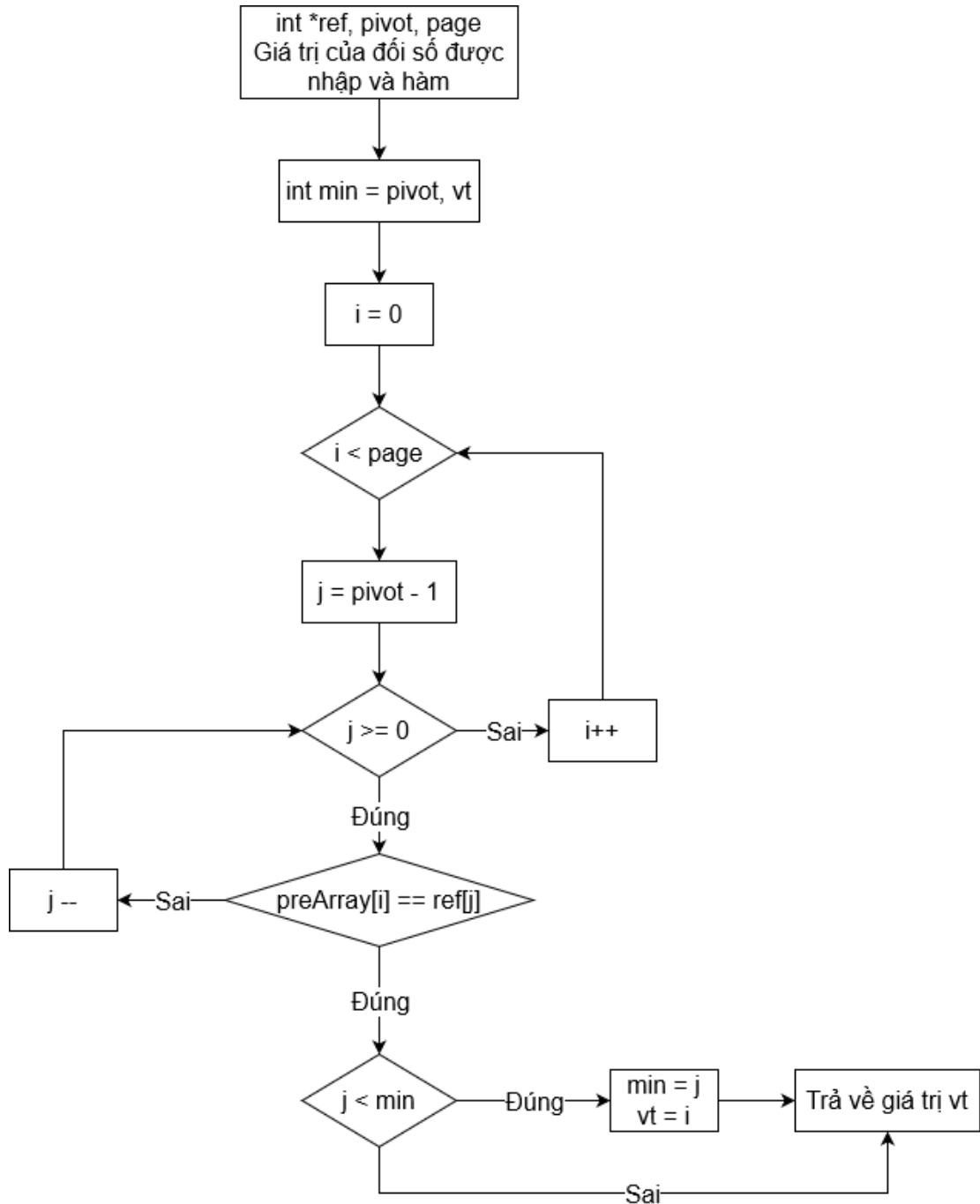
the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 20
Nhap danh sach trang: 6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
3
--- Page Replacement algorithm---
6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1
6 6 6 6 6 6 3 3 3 3 3 3 5 6 6 6 6 3 5 1
0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
0 0 0 0 5 5 5 1 1 1 1 7 7 7 7 7 7 7 7 7
* * * * *
Number of Page Fault: 12

```

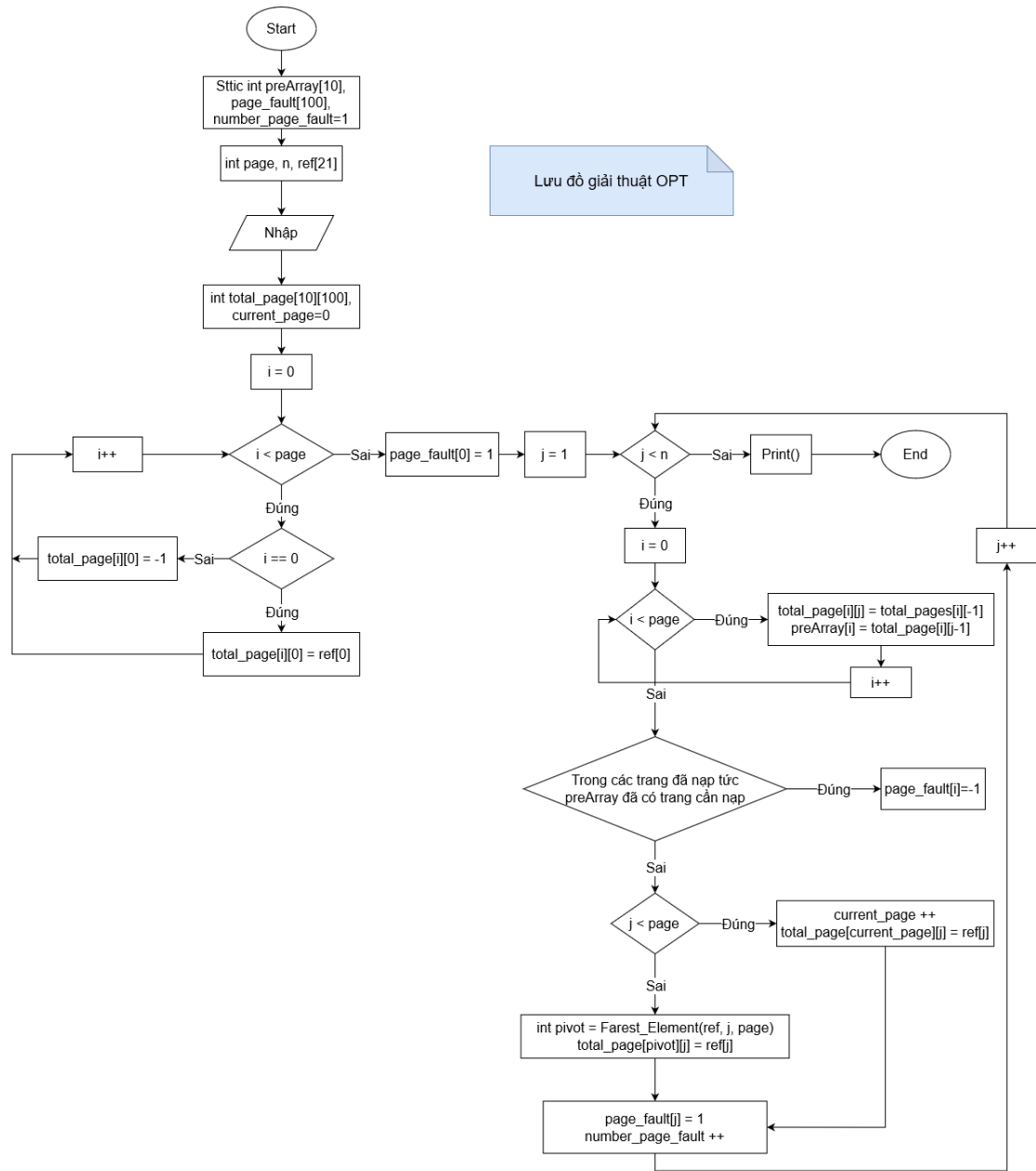
Hình 9: Kết quả giải ví dụ 3 bằng thuật toán LRU

### 3. Task name 3: Giải thuật OPT

#### 📌 Lưu đồ thuật toán



Hình 10: Lưu đồ hàm tìm ra vị trí cần được thay thế.



Hình 11: Lưu đồ thuật toán OPT.

### Giải thích

- Hình 10: Hàm sẽ tiến hành duyệt qua tất cả các giá trị trong cột và đi tìm vị trí trong mảng của mảng ref[] để tìm ra trang sẽ được gọi lại muộn nhất trong tương lai.
- Hình 11: Tương tự như các bước của thuật toán FIFO, ở thuật toán OPT tại bước 5 để xác định vị trí page cần thay để ta sẽ tiến hành gọi hàm

**Farest\_Element** (xác định vị trí trang cần thay thế) rồi tất cả mọi bước thì được thực hiện như FIFO.

## Test Case

- Ví dụ 1: Xét chuỗi truy xuất bộ nhớ sau: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế OPT, giả sử có 5 khung trang?

+ Lời giải:

Giải bằng tay:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	6	6	6	6	6	6	6	6
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
*	*	*	*			*	*					*							

=> Tổng cộng có 7 lỗi trang.

Giải bằng code:

```

the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
Nhập số lượng: 20
Nhập danh sách trang: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
--- Page Replacement algorithm ---
Input page frames: 5
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
2
--- Page Replacement algorithm---
1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 6 6 6 6 6 7 7 7 7 1 1 1 1 1 1 1
* * * * *
Number of Page Fault: 10

```

Hình 12: Kết quả giải ví dụ 1 bằng thuật toán OPT

- Ví dụ 2: Xét chuỗi truy xuất bộ nhớ sau: 1, 3, 4, 5, 2, 3, 6, 3. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế OPT, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

1	3	4	5	2	3	6	3
1	1	1	1	2	2	2	2
	3	3	3	3	3	3	3
		4	4	4	4	6	6
			5	5	5	5	5
*	*	*	*	*		*	

=> Tổng cộng có 6 lỗi trang.

Giải bằng code:

```

the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 8
Nhập danh sách trang: 1 3 4 5 2 3 6 3
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
2
--- Page Replacement algorithm---
1 3 4 5 2 3 6 3
1 1 1 1 2 2 2 2
  3 3 3 3 3 3
    4 4 4 6 6
      5 5 5 5 5
* * * * *
Number of Page Fault: 6

```

Hình 13: Kết quả giải ví dụ 2 bằng thuật toán OPT

- Ví dụ 3: Xét chuỗi truy xuất bộ nhớ sau: 6, 2, 4, 4, 5, 6, 3, 1, 4, 2, 3, 7, 5, 6, 7, 2, 4, 3, 5, 1. Có bao nhiêu lỗi trang xảy ra khi sử dụng thuật toán thay thế OPT, giả sử có 4 khung trang?

+ Lời giải:

Giải bằng tay:

6	2	4	4	5	6	3	1	4	2	3	7	5	6	7	2	4	3	5	1
6	6	6	6	6	6	3	3	3	3	3	3	5	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3
		4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5
				5	5	5	1	1	1	1	7	7	7	7	7	7	7	7	1
*	*	*		*		*	*				*	*	*				*	*	*

=> Tổng cộng có 12 lỗi trang.

Giải bằng code:

```

the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ g++ Lab06.cpp -o Lab06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 20
Nhập danh sách trang: 6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
2
--- Page Replacement algorithm---
6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 2 4 3 5 1
6 6 6 6 5 5 5 1 1 1 3 3 3 6 6 6 4 4 4 1
  2 2 2 2 6 6 6 4 4 4 7 7 7 7 7 7 3 3 3
    4 4 4 4 3 3 3 2 2 2 5 5 5 2 2 2 5 5
***
Number of Page Fault: 18

```

Hình 14: Kết quả giải ví dụ 3 bằng thuật toán OPT

## Soucre Code Của Chương Trình

```
1 /*#####
2 # University of Information Technology
3 # IT007 Operating System
4 # Pham Duc The, 19522253
5 # File: Lab06.cpp
6 #####*/
7
8
9 #include<iostream>
10 using namespace std;
11 static int preArray[10];
12 static int page_fault[100];
13 static int number_page_fault = 1;
14
15 int Is_in_preArray(int page, int value) {
16     for (int i = 0; i < page; i++) {
17         if (value == preArray[i]) return i;
18     }
19     return -1;
20 }
21
22
23 int Farest_Element(int *ref, int pivot, int page) {
24     int min = pivot;
25     int vt;
26     for (int i = 0; i < page; i++) {
27         for (int j = pivot - 1; j >= 0; j--) {
28             if (preArray[i] == ref[j]) {
29                 if (j < min) {
30                     min = j;
31                     vt = i;
32                 }
33             }
34             break;
35         }
36     }
37 }
38 }
39 return vt;
40 }
41
42 int Farest_Element_Oppsote(int *ref, int pivot, int page, int n) {
43     int max = pivot;
44     int flag[10];
45     for (int i = 0; i < page; i++) {
46         flag[i] = false;
47     }
48     int vt = -1;
49     for (int i = 0; i < page; i++) {
50         for (int j = pivot + 1; j < n; j++) {
51             if (preArray[i] == ref[j]) {
52                 if (j > max) {
53                     max = j;
54                     vt = i;
55                 }
56                 flag[i] = true;
57                 break;
58             }
59         }
60     }
61 }
62
63 for (int i = 0; i < page; i++) {
64     if (flag[i] == false) return i;
65 }
66 return vt;
67 }
68
```

Hình 15: Soucre code chương trình từ dòng 0-68



```

68
69 void Print(int total_page[10][100], int n, int page, int ref[100]) {
70     // Print
71     for (int i = 0; i < n; i++) {
72         cout << ref[i] << " ";
73     }
74     cout << endl;
75     for (int i = 0; i < page; i++) {
76         for (int j = 0; j < n; j++) {
77             if (total_page[i][j] != -1) {
78                 cout << total_page[i][j] << " ";
79             }
80             else {
81                 cout << " ";
82             }
83         }
84         cout << endl;
85     }
86     for (int i = 0; i < n; i++) {
87         if (page_fault[i] == 1) cout << "** ";
88         else {
89             cout << " ";
90         }
91     }
92     cout << endl;
93     cout << "Number of Page Fault: " << number_page_fault << endl;
94 }
95
96 void FIFO(int ref[], int n, int page)
97 {
98     bool IsFault;
99     int total_page[10][100];
100     int current_page = 0;
101     for (int i = 0; i < page; i++) {
102         if (i == 0) { total_page[i][0] = ref[0]; }
103         else {
104             total_page[i][0] = -1;
105         }
106     }
107     page_fault[0] = 1;
108     for (int j = 1; j < n; j++) {
109         for (int i = 0; i < page; i++) {
110             total_page[i][j] = total_page[i][j-1];
111             preArray[i] = total_page[i][j - 1];
112         }
113         if (Is_in_preArray(page, ref[j]) != -1) {
114             page_fault[j] = -1;
115         }
116         else {
117             current_page++;
118             if (current_page == page) current_page = 0;
119             total_page[current_page][j] = ref[j];
120             page_fault[j] = 1;
121             number_page_fault++;
122         }
123     }
124 }
125
126 Print(total_page, n, page, ref);
127 }
128
129

```

Hình 16: Soucre code chương trình từ dòng 69-129

```

130 void OPT(int ref[], int n, int page)
131 {
132     bool IsFault;
133     int total_page[10][100];
134     int current_page = 0;
135     for (int i = 0; i < page; i++) {
136         if (i == 0) { total_page[i][0] = ref[0]; }
137         else {
138             total_page[i][0] = -1;
139         }
140     }
141     page_fault[0] = 1;
142
143     for (int j = 1; j < n; j++)
144     {
145
146         for (int i = 0; i < page; i++) {
147             total_page[i][j] = total_page[i][j - 1];
148             preArray[i] = total_page[i][j - 1];
149         }
150         /////
151         if (Is_in_preArray(page, ref[j]) != -1) {
152             page_fault[j] = 0;
153         }
154         else {
155             if (j < page) {
156                 current_page++;
157                 total_page[current_page][j] = ref[j];
158             }
159             else {
160
161                 int pivot = Fareast_Element(ref, j, page);
162                 total_page[pivot][j] = ref[j];
163             }
164             page_fault[j] = 1;
165             number_page_fault++;
166         }
167         /////
168     }
169     Print(total_page, n, page, ref);
170 }
171
172 void LRU(int ref[], int n, int page)
173 {
174     bool IsFault;
175     int total_page[10][100];
176     int current_page = 0;
177     for (int i = 0; i < page; i++) {
178         if (i == 0) { total_page[i][0] = ref[0]; }
179         else {
180             total_page[i][0] = 0;
181         }
182     }
183     page_fault[0] = 1;
184
185     for (int j = 1; j < n; j++)
186     {
187
188         for (int i = 0; i < page; i++) {
189             total_page[i][j] = total_page[i][j - 1];
190             preArray[i] = total_page[i][j - 1];
191         }
192         /////
193         if (Is_in_preArray(page, ref[j]) != -1) {
194             page_fault[j] = 0;
195         }

```

Hình 17: Soucre code chương trình từ dòng 130-195

```

196         else {
197             if (j < page) {
198                 current_page++;
199                 total_page[current_page][j] = ref[j];
200             }
201             else {
202
203                 int pivot = Farest_Element_Opposite(ref, j, page, n);
204                 total_page[pivot][j] = ref[j];
205             }
206             page_fault[j] = 1;
207             number_page_fault++;
208         }
209         ////
210     }
211     Print(total_page, n, page, ref);
212 }
213
214
215 int main() {
216     int page; int temp;
217     int ref[11] = { 1, 7, 5, 2, 0, 4, 3, 3, 0, 0, 7 };
218     int n = 11;
219     cout << "--- Page Replacement algorithm ---" << endl;
220     cout << "1. Default referenced sequence" << endl;
221     cout << "2. Manual input sequence" << endl;
222     cin >> temp;
223     switch (temp) {
224     case 1:
225
226         break;
227     case 2:
228         cout << "Nhap so luong: "; cin >> n;
229         cout << "Nhap danh sach trang: ";
230         for (int i = 0; i < n; i++) {
231             cin >> ref[i];
232         }
233     }
234
235
236
237
238
239     cout << "--- Page Replacement algorithm ---" << endl;
240     cout << "Input page frames: "; cin >> page;
241     cout << "--- Select algorithm ---" << endl;
242     cout << "1. FIFO algorithm" << endl;
243     cout << "2. OPT algorithm" << endl;
244     cout << "3. LRU algorithm" << endl;
245     cout << "--- Enter input ---" << endl;
246
247     cin >> temp;
248     cout << "--- Page Replacement algorithm--- " << endl;
249     switch (temp) {
250     case 1:
251         FIFO(ref, n, page);
252         break;
253     case 2:
254         OPT(ref, n, page);
255         break;
256     case 3:
257         LRU(ref, n, page);
258         break;
259     }
260
261     system("pause");
262     return 0;
263 }
264

```

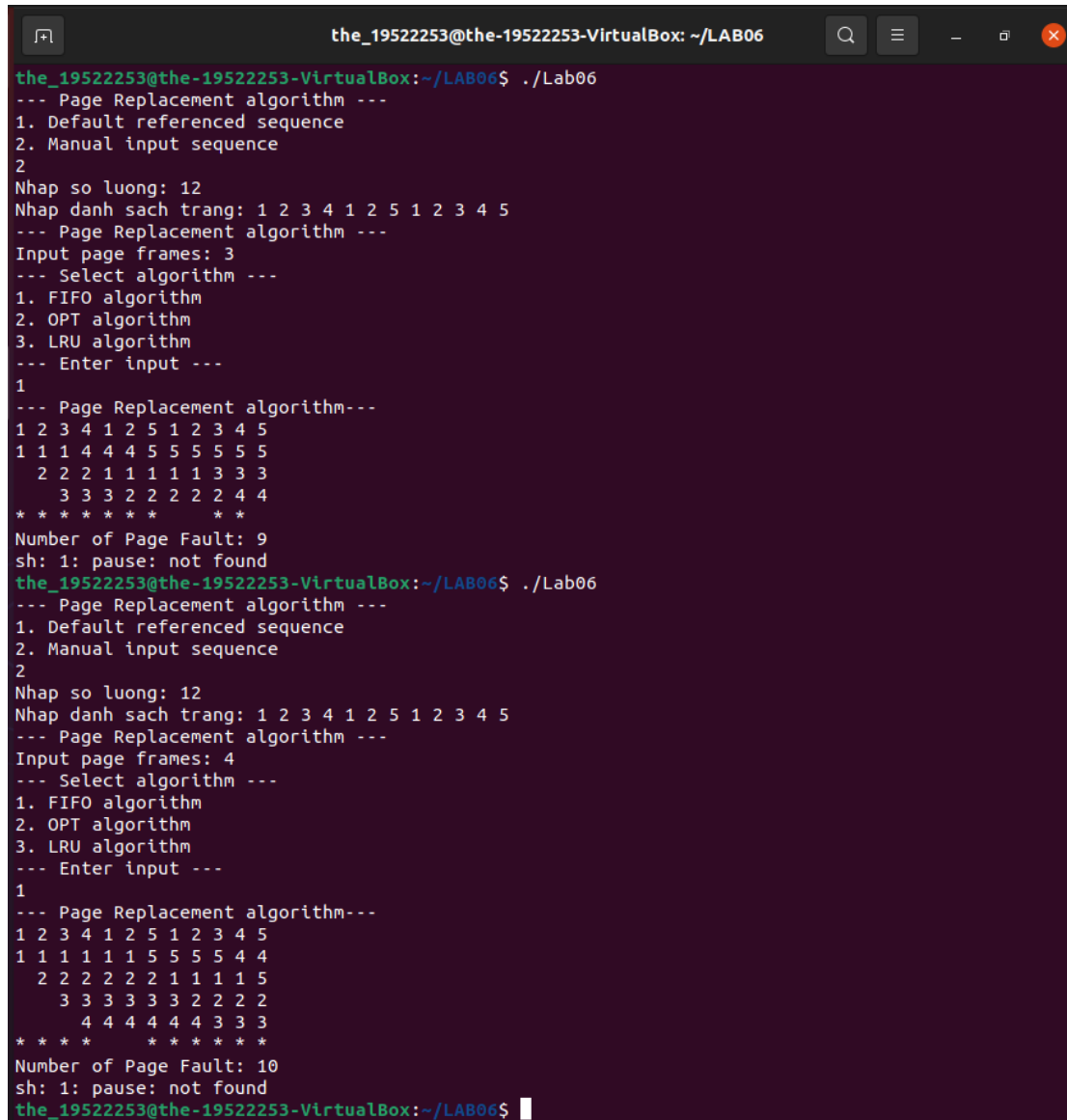
Hình 18: Soucre code chương trình từ dòng 196-264

## Section 6.5

1. Task name 1: Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.

🚦 Nghịch lý Belady là hiện tượng số lỗi trang tăng lên khi tăng số frame.

🚦 Chứng minh: Với chuỗi 1 2 3 4 1 2 5 1 2 3 4 5 (12 phần tử) và thuật toán FIFO ta có:



```
the_19522253@the-19522253-VirtualBox: ~/LAB06
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames: 3
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 4 4 4 5 5 5 5 5 5
  2 2 1 1 1 1 1 3 3 3
    3 3 3 2 2 2 2 2 4 4
* * * * *
Number of Page Fault: 9
sh: 1: pause: not found
the_19522253@the-19522253-VirtualBox:~/LAB06$ ./Lab06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 12
Nhap danh sach trang: 1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames: 4
--- Select algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
--- Enter input ---
1
--- Page Replacement algorithm---
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 1 1 1 5 5 5 5 4 4
  2 2 2 2 2 1 1 1 1 5
    3 3 3 3 3 2 2 2 2
      4 4 4 4 4 3 3 3
* * * * *
Number of Page Fault: 10
sh: 1: pause: not found
the_19522253@the-19522253-VirtualBox:~/LAB06$
```


Hình 19: Chứng minh nghịch lý Belady


Với 3 frame có 9 lỗi trang, 4 frame lại có tới 10 lỗi trang.

## 2. Task name 2: Nhận xét về mức độ hiệu quả và tính khả thi của các giải thuật FIFO, OPT, LRU.

 Nhận xét:

- 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 bất khả thi nhất là OPT vì việc biết trước những trang nào có thể được truy xuất tiếp theo gần như là điều không thể.

 Giải thuật phức tạp nhất là OPT và LRU 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.