

Họ và tên: Đặng Thiên Ân

MSSV: 23520003

Lớp: IT007.P110.2

HỆ ĐIỀU HÀNH

BÁO CÁO LAB 6

Tiêu Đề: 23520003-LAB6

Nội dung:

Section 6.4:

Task 1> Xong

Task 2> Xong

Task 3> Xong

Section 6.5:

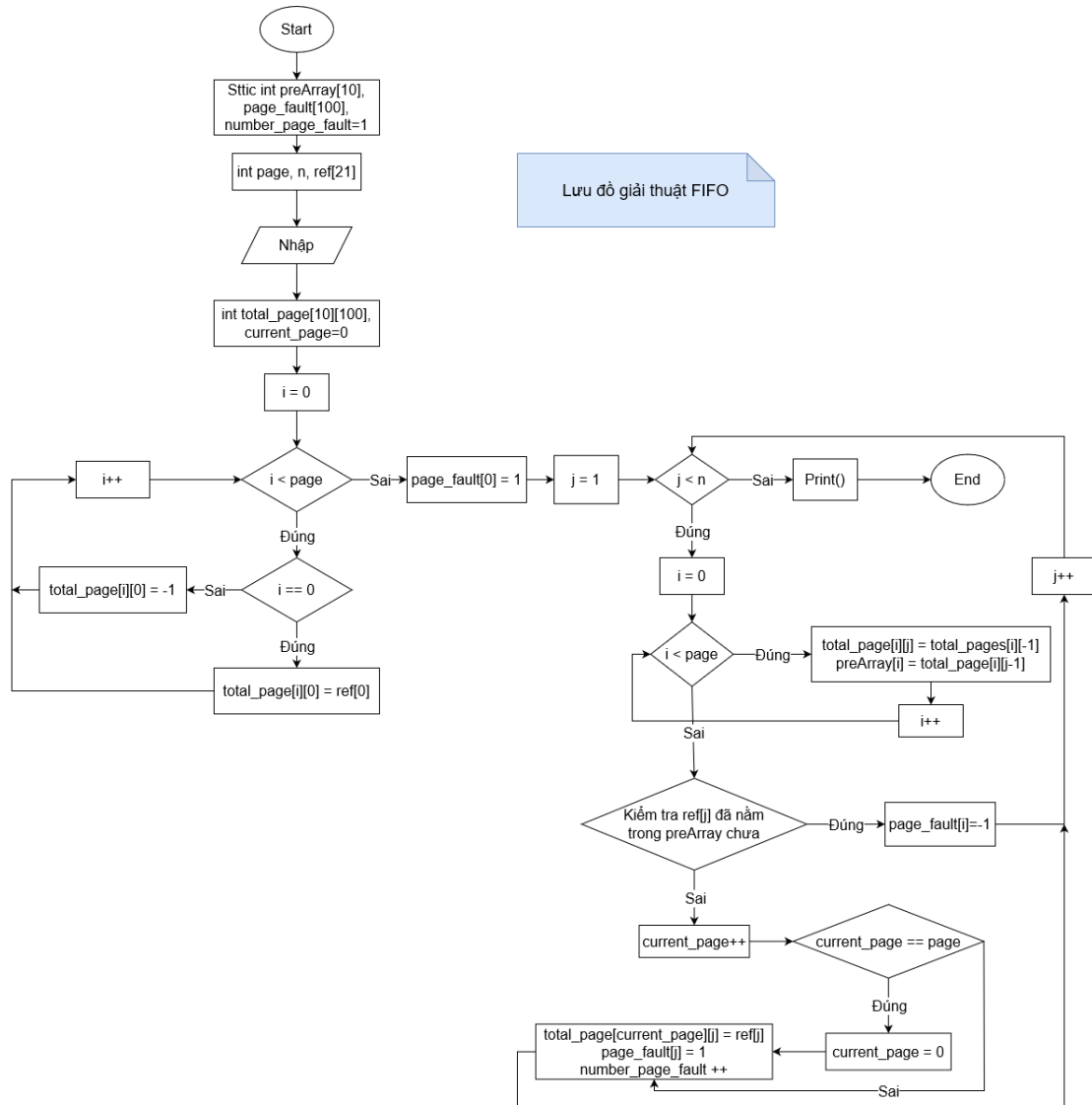
Task 1> Xong

Task 2> Xong

Section 6.4

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 `preArry[]`, 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:

```

pentakll4002@123123: ~
pentakll4002@123123:~$ gedit LAB06.c
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./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 6
  2 2 2 2 2 2 2 2 1 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 2
      4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3
        5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 7
* * * * *
Number of Page Fault: 10
sh: 1: pause: not found
*** stack smashing detected ***: terminated
Aborted (core dumped)

```

Hình 3: 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:

```

pentakll4002@123123: ~
pentakll4002@123123:~$ gedit LAB06.c
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./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
* * * * *
Number of Page Fault: 7

```

Hình 4: 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:

```

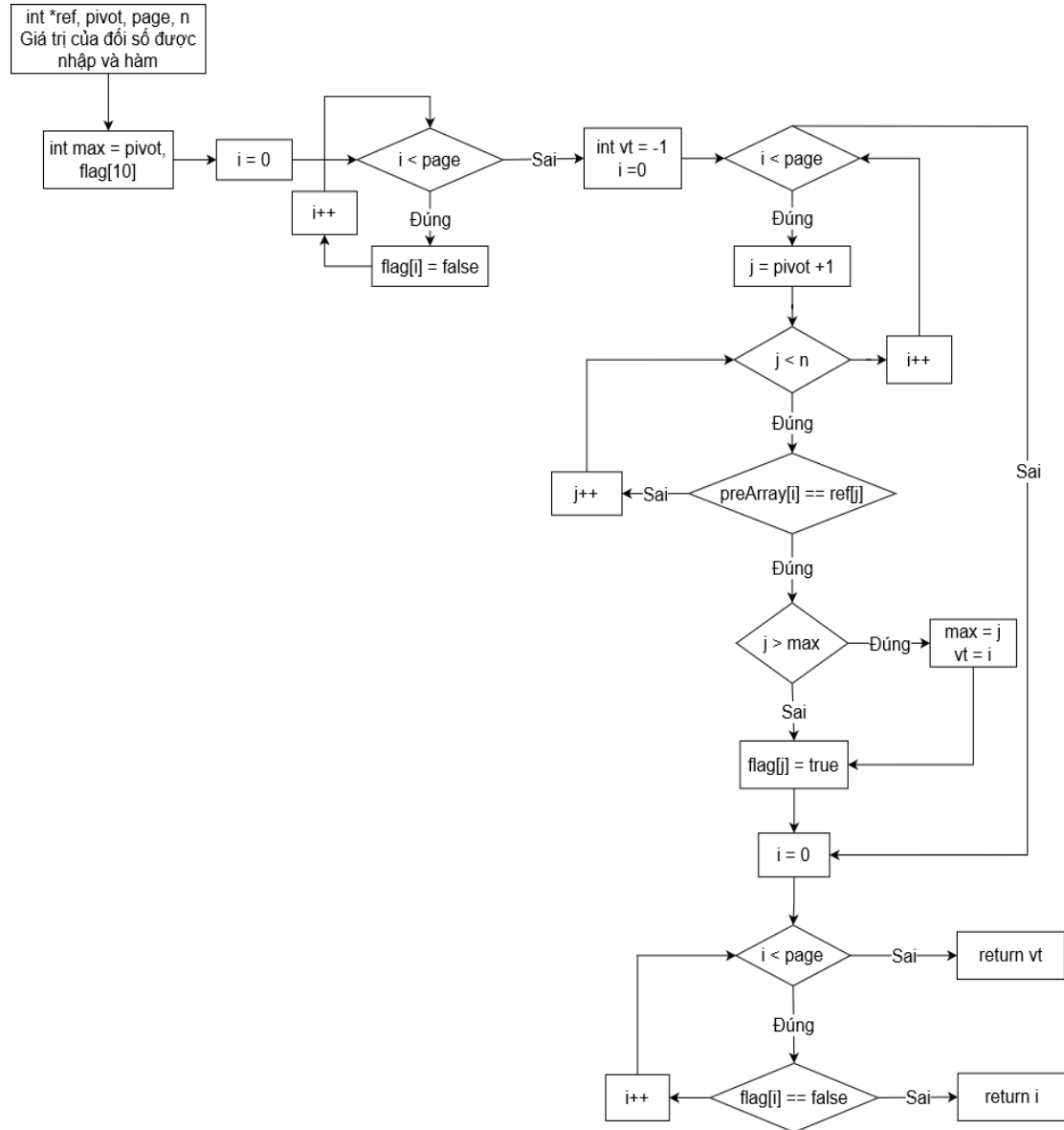
Number of Page Fault: 7
sh: 1: pause: not found
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./LAB06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhap so luong: 20
Nhap danh sach trang: 6 2 4 4 5 6 3 1 4 2 3 7 5 6 7 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 3 3 3
* * * * *
Number of Page Fault: 13
sh: 1: pause: not found

```

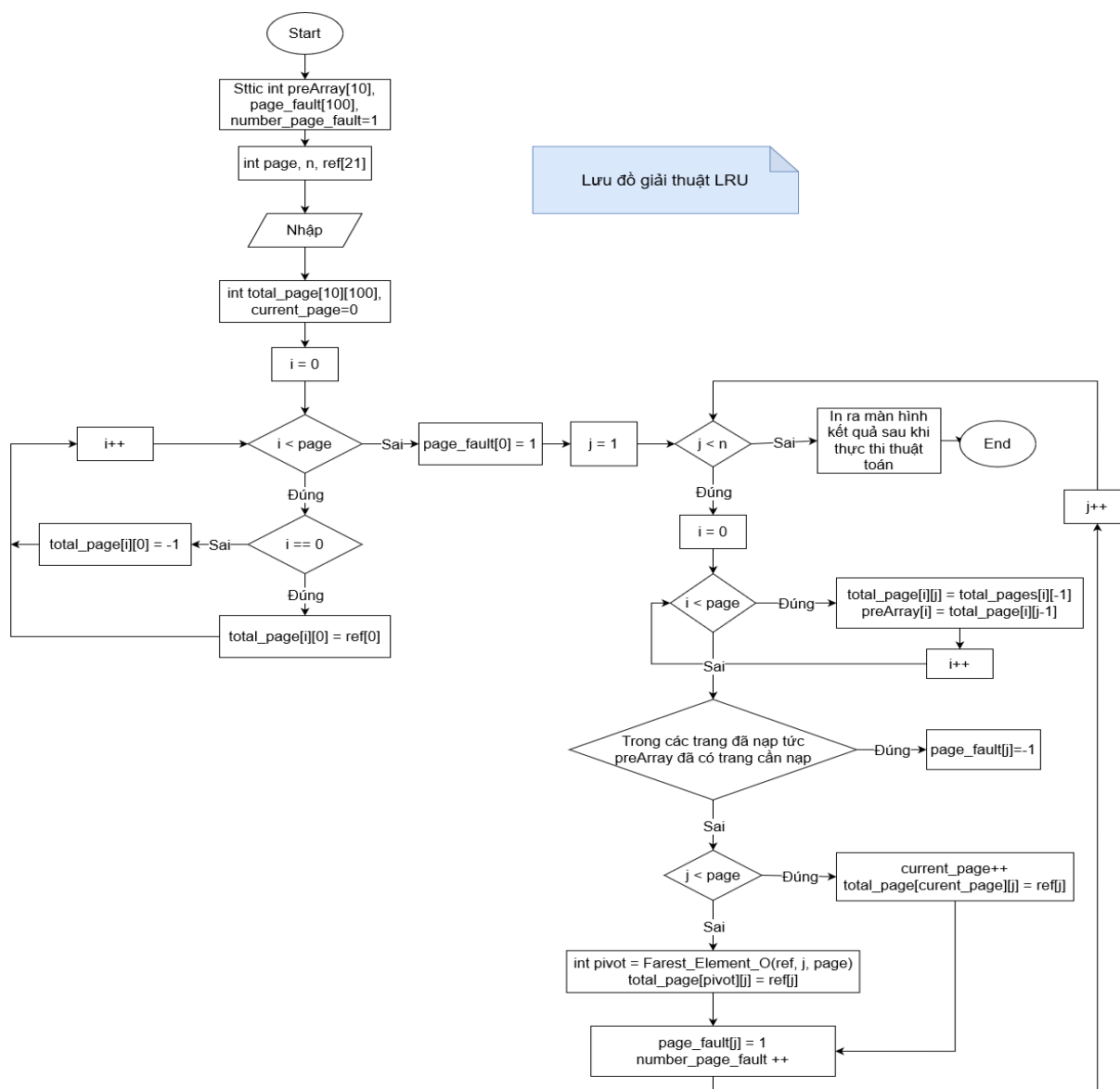
Hình 5: Kết quả giải ví dụ 3 bằng thuật toán FIFO

2. Task name 2: Giải thuật LRU

– Lưu đồ thuật toán



Hình 6: Lưu đồ hàm tìm ra trang được gọi sớm nhất trong quá khứ



Hình 7: 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:

```

pentakll4002@123123: ~
Aborted (core dumped)
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./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 ---
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 0 7 7 7 7 7 7 7 7
* * * * *
Number of Page Fault: 7
sh: 1: pause: not found

```

Hình 8: 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:

```

Number of Page Fault: 7
sh: 1: pause: not found
*** stack smashing detected ***: terminated
Aborted (core dumped)
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./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 9: 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	5	4	4	4
*	*	*		*		*	*	*	*		*	*	*		*	*	*	*	*

=> Tổng cộng có 16 lỗi trang.

Giải bằng code:

```

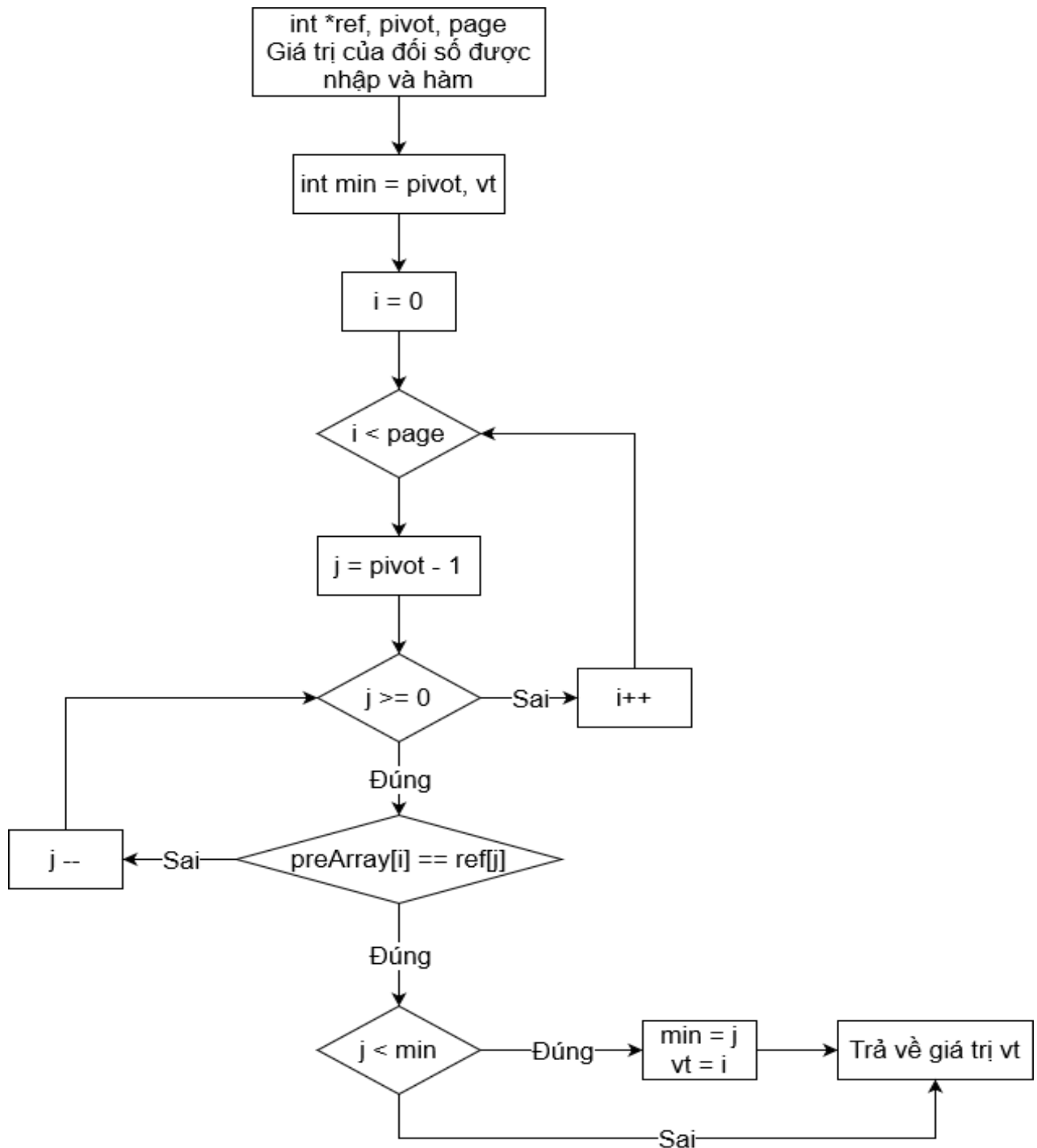
pentakll4002@123123: ~
Number of Page Fault: 6
sh: 1: pause: not found
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./LAB06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 20
Nhập danh sách trang: 6 2 4 4 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 ---
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 6 3 5 1
0 2 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 4
0 0 0 0 5 5 5 1 1 1 1 7 7 7 7 7 7 7 7 7 7
* * * * *
Number of Page Fault: 12
sh: 1: pause: not found

```

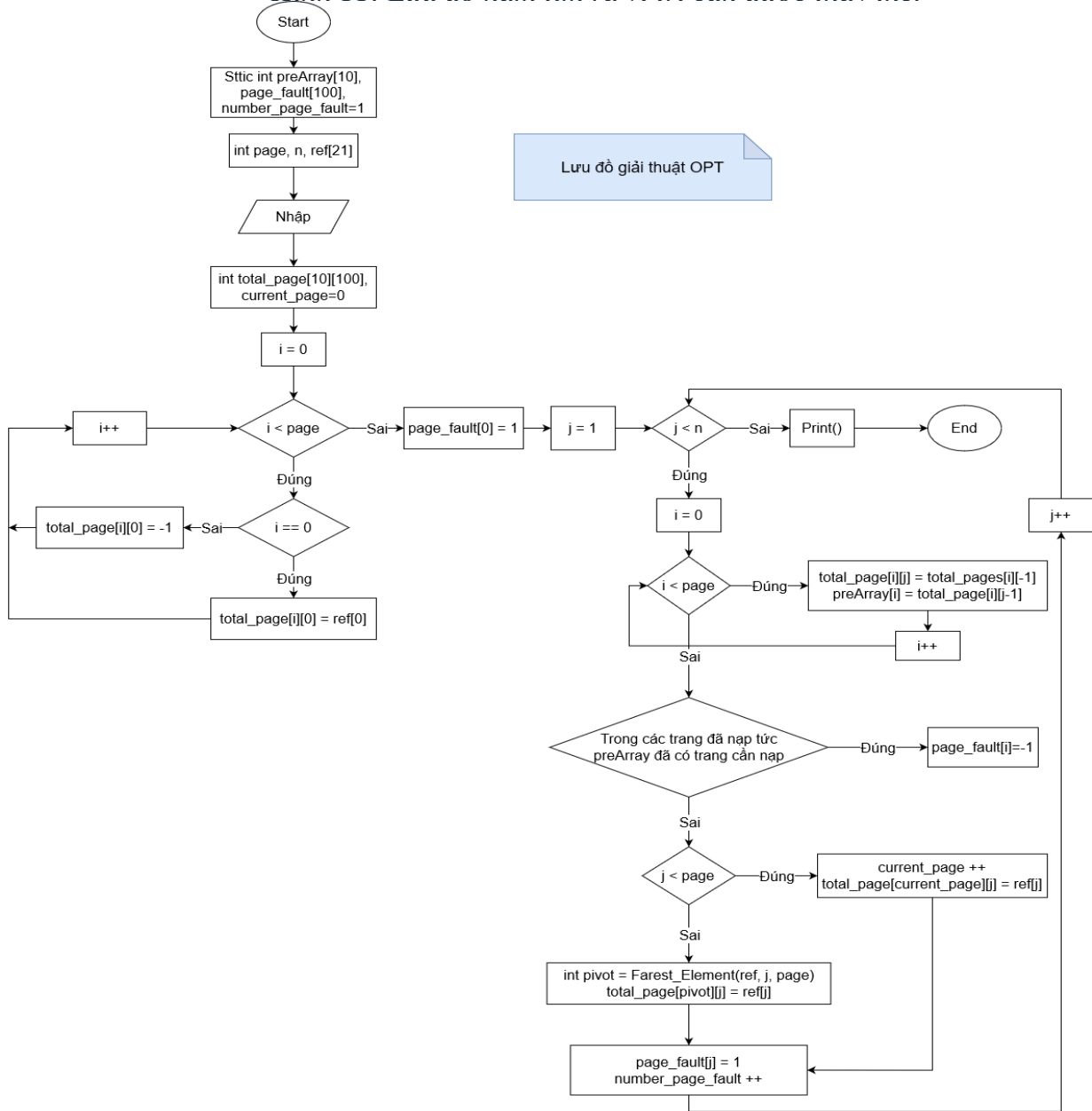
Hình 10: Kết quả giải ví dụ 3 bằng thuật toán LRU

3. Task name 3: Giải thuật OPT

– Lưu đồ thuật toán



Hình 11: Lưu đồ hàm tìm ra vị trí cần được thay thế.



Hình 12: 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 thế 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:

```

pentakll4002@123123: ~
Aborted (core dumped)
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./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 ---
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 6 6 6 6 6 6 6 6
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
ch: 1: pause: not found

```

Hình 13: 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:

```

pentakll4002@123123: ~
Number of Page Fault: 10
sh: 1: pause: not found
*** stack smashing detected ***: terminated
Aborted (core dumped)
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./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 3 3
4 4 4 4 6 6
5 5 5 5 5
* * * * *
Number of Page Fault: 6
sh: 1: pause: not found

```

Hình 14: 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:

```

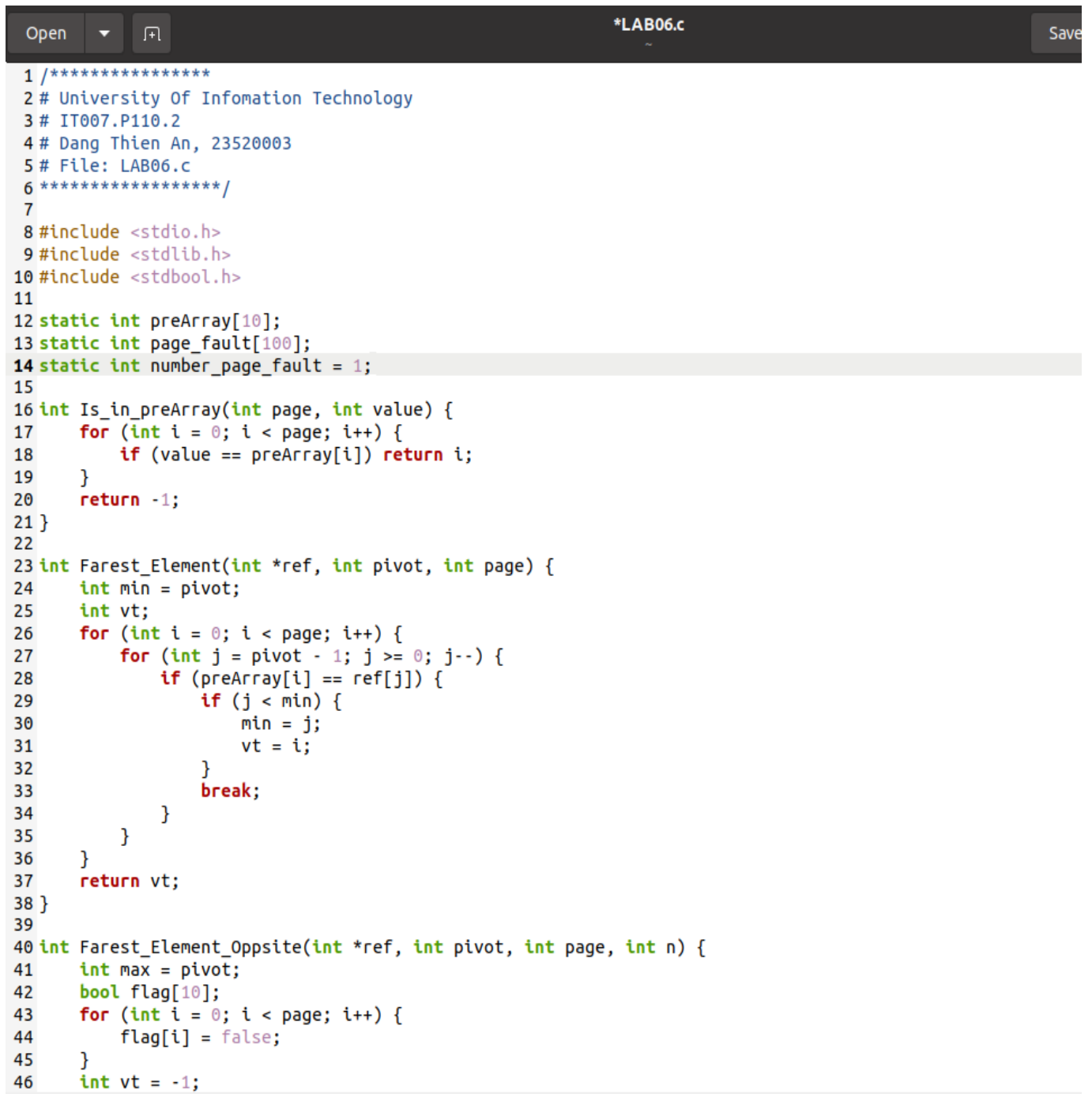
pentakll4002@123123: ~
Number of Page Fault: 6
sh: 1: pause: not found
pentakll4002@123123:~$ gcc LAB06.c -o LAB06 -lpthread -lrt
pentakll4002@123123:~$ ./LAB06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 20
Nhập danh sách trang: 6 2 4 4 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
sh: 1: pause: not found

```

Hình 15: Kết quả giải ví dụ 3 bằng thuật toán OPT

– Soucre Code Của Chương Trình:



```
1 /*****
2 # University Of Infomation Technology
3 # IT007.P110.2
4 # Dang Thien An, 23520003
5 # File: LAB06.c
6 *****/
7
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <stdbool.h>
11
12 static int preArray[10];
13 static int page_fault[100];
14 static int number_page_fault = 1;
15
16 int Is_in_preArray(int page, int value) {
17     for (int i = 0; i < page; i++) {
18         if (value == preArray[i]) return i;
19     }
20     return -1;
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         }
35     }
36     return vt;
37 }
38
39 int Farest_Element_Oppsite(int *ref, int pivot, int page, int n) {
40     int max = pivot;
41     bool flag[10];
42     for (int i = 0; i < page; i++) {
43         flag[i] = false;
44     }
45     int vt = -1;
```

Hình 16: Source code chương trình từ dòng 0-46

```

Open  ▾  [icon]  *LAB06.c
46  int vt = -1;
47  for (int i = 0; i < page; i++) {
48      for (int j = pivot + 1; j < n; j++) {
49          if (preArray[i] == ref[j]) {
50              if (j > max) {
51                  max = j;
52                  vt = i;
53              }
54              flag[i] = true;
55              break;
56          }
57      }
58  }
59
60  for (int i = 0; i < page; i++) {
61      if (!flag[i]) return i;
62  }
63  return vt;
64 }
65
66 void Print(int total_page[10][100], int n, int page, int ref[100]) {
67     // Print
68     for (int i = 0; i < n; i++) {
69         printf("%d ", ref[i]);
70     }
71     printf("\n");
72     for (int i = 0; i < page; i++) {
73         for (int j = 0; j < n; j++) {
74             if (total_page[i][j] != -1) {
75                 printf("%d ", total_page[i][j]);
76             } else {
77                 printf(" ");
78             }
79         }
80         printf("\n");
81     }
82     for (int i = 0; i < n; i++) {
83         if (page_fault[i] == 1) printf("* ");
84         else {
85             printf(" ");
86         }
87     }
88     printf("\n");
89     printf("Number of Page Fault: %d\n", number_page_fault);
90 }
91

```

Hình 16: Source code chương trình từ dòng 46-90

```
Open  ▾  [icon]  *LAB06.c

91
92 void FIFO(int ref[], int n, int page) {
93     bool IsFault;
94     int total_page[10][100];
95     int current_page = 0;
96     for (int i = 0; i < page; i++) {
97         if (i == 0) { total_page[i][0] = ref[0]; }
98         else {
99             total_page[i][0] = -1;
100         }
101     }
102     page_fault[0] = 1;
103
104     for (int j = 1; j < n; j++) {
105         for (int i = 0; i < page; i++) {
106             total_page[i][j] = total_page[i][j-1];
107             preArray[i] = total_page[i][j - 1];
108         }
109         if (Is_in_preArray(page, ref[j]) != -1) {
110             page_fault[j] = -1;
111         } else {
112             current_page++;
113             if (current_page == page) current_page = 0;
114             total_page[current_page][j] = ref[j];
115             page_fault[j] = 1;
116             number_page_fault++;
117         }
118     }
119     Print(total_page, n, page, ref);
120 }
121
122 void OPT(int ref[], int n, int page) {
123     bool IsFault;
124     int total_page[10][100];
125     int current_page = 0;
126     for (int i = 0; i < page; i++) {
127         if (i == 0) { total_page[i][0] = ref[0]; }
128         else {
129             total_page[i][0] = -1;
130         }
131     }
132     page_fault[0] = 1;
133
134     for (int j = 1; j < n; j++) {
135         for (int i = 0; i < page; i++) {
136             total_page[i][j] = total_page[i][j - 1];
```

Hình 17: Source code chương trình từ dòng 91-136

```
Open ▾ [icon] *LAB06.c
137     preArray[i] = total_page[i][j - 1];
138 }
139 if (Is_in_preArray(page, ref[j]) != -1) {
140     page_fault[j] = 0;
141 } else {
142     if (j < page) {
143         current_page++;
144         total_page[current_page][j] = ref[j];
145     } else {
146         int pivot = Fareast_Element(ref, j, page);
147         total_page[pivot][j] = ref[j];
148     }
149     page_fault[j] = 1;
150     number_page_fault++;
151 }
152 }
153 Print(total_page, n, page, ref);
154 }
155
156 void LRU(int ref[], int n, int page) {
157     bool IsFault;
158     int total_page[10][100];
159     int current_page = 0;
160     for (int i = 0; i < page; i++) {
161         if (i == 0) { total_page[i][0] = ref[0]; }
162         else {
163             total_page[i][0] = 0;
164         }
165     }
166     page_fault[0] = 1;
167
168     for (int j = 1; j < n; j++) {
169         for (int i = 0; i < page; i++) {
170             total_page[i][j] = total_page[i][j - 1];
171             preArray[i] = total_page[i][j - 1];
172         }
173         if (Is_in_preArray(page, ref[j]) != -1) {
174             page_fault[j] = 0;
175         } else {
176             if (j < page) {
177                 current_page++;
178                 total_page[current_page][j] = ref[j];
179             } else {
180                 int pivot = Fareast_Element_Oppsite(ref, j, page, n);
181                 total_page[pivot][j] = ref[j];
182             }
183         }
184     }
185 }
```

Hình 18: Source code chương trình từ dòng 136-181

```

181         total_page[pivot][j] = ref[j];
182     }
183     page_fault[j] = 1;
184     number_page_fault++;
185 }
186 }
187 Print(total_page, n, page, ref);
188 }
189
190 int main() {
191     int page, temp;
192     int ref[11] = { 1, 7, 5, 2, 0, 4, 3, 3, 0, 0, 7 };
193     int n = 11;
194     printf("--- Page Replacement algorithm ---\n");
195     printf("1. Default referenced sequence\n");
196     printf("2. Manual input sequence\n");
197     scanf("%d", &temp);
198     switch (temp) {
199     case 1:
200         break;
201     case 2:
202         printf("Nhap so luong: ");
203         scanf("%d", &n);
204         printf("Nhap danh sach trang: ");
205         for (int i = 0; i < n; i++) {
206             scanf("%d", &ref[i]);
207         }
208     }
209
210     printf("--- Page Replacement algorithm ---\n");
211     printf("Input page frames: ");
212     scanf("%d", &page);
213     printf("--- Select algorithm ---\n");
214     printf("1. FIFO algorithm\n");
215     printf("2. OPT algorithm\n");
216     printf("3. LRU algorithm\n");
217     printf("--- Enter input ---\n");
218     scanf("%d", &temp);
219     printf("--- Page Replacement algorithm--- \n");
220     switch (temp) {
221     case 1:
222         FIFO(ref, n, page);
223         break;
224     case 2:
225         OPT(ref, n, page);
226         break;

```

Hình 19: Source code chương trình từ dòng 181-225

```

224     case 2:
225         OPT(ref, n, page);
226         break;
227     case 3:
228         LRU(ref, n, page);
229         break;
230 }
231

```

Hình 19.1: Source code chương trình từ dòng 225-230

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:

```

pentakll4002@123123: ~
pentakll4002@123123:~$ ./LAB06
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
Nhập số lượng: 12
Nhập danh sách 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 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
pentakll4002@123123:~$ ./LAB06

```

Hình 20: Chứng minh nghịch lý Belady

```

pentak114002@123123:~$ ./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 2 1 1 1 1 5
    3 3 3 3 3 3 2 2 2 2
      4 4 4 4 4 4 3 3 3
* * * * * * * * * *
Number of Page Fault: 10
sh: 1: pause: not found

```

Hình 20: 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.

