**N19DCCN049 – Nguyễn Anh Hào**

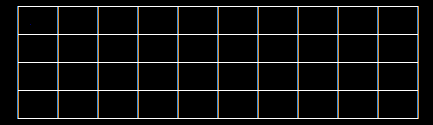
**Đề tài**: Viết chương trình mô phỏng các giải thuật thay thế trang theo thuật toán FIFO

Chương trình gồm có:

* Cấu trúc ToaDo (Toạ độ)
* Hàm:
* Một số hàm có sẵn lấy từ các thư viện Windows.h:
* resizeConsole(int width, int height): đặt lại kích thước console.
* ShowCur(bool CursorVisibility): hiện thị con trỏ.
* setFontSize(int FontSize): chọn kích thước font chữ.
* init(int \*arr, int n): gán giá trị -1 cho tất cả các phần tử có trong mảng.
* nhap(int \*arr, int n): nhập giá trị cho các phần tử trong mảng.
* isExists(int \*arr, int n, int value): kiểm tra xem trong mảng có phần tử nào có giá trị bằng value hay không, nếu có thì trả về true, ngược lại trả về false.
* banner(): đọc file banner.txt và hiển thị tên thuật toán ra màn hình.
* outChar(int x, int y, char data): di chuyển vị trí con trỏ đến vị trí có toạ độ A(x, y) và in ra màn hình 1 kí tự.
* line(ToaDo start, int length, int character): in số lượng (length) kí tự (char) bắt đầu từ vị trí có toạ độ start để tạo thành 1 line.
* colorBox(ToaDo A, ToaDo C, string text, int Bgcolor, int textcolor): Tạo ra một box từ toạ độ điểm A và C, tô màu nền (BGcolor), chọn màu text (textcolor) và đặt text vào giữa box.
* xoaNoiDung(int y, int sodong, int color): in ra các dòng là dãy các khoảng trắng để xoá nội đã in trên màn hình console và có thể chọn màu sao cho phù hợp với vị trí cần xoá.
* thongTinThuatToan(): trong hàm này có gọi lại hàm colorBox và cấu trúc ToaDo để tạo ra các box chứa tiêu đề về các thông tin của thuật toán như: số trang(number of pages), danh sách trang(pages), số khung trang(number of frame) và box để nhập dữ liệu.



* input(int \*pages, int &soTrang, int &soKhungTrang): sử dụng tham chiếu làm tham số để khi nhập giá trị thì sẽ thay đổi giá trị của tham số đó bên ngoài hàm. Trong thuật toán này có sử dụng hàm ShowCur(true)(lấy từ thư viện Windows.h), mục đích để tạo hiệu ứng nhấp nháy con trỏ để giao diện nhập dữ liệu sinh động hơn.(trước đó đã gọi hàm ShowCur(false) để tránh lỗi giao diện)
* drawBorder(int soTrang, int soKhungTrang): trong hàm này có gọi lại các hàm outChar, line và sử dụng các vòng lặp for để in ra các kí tự đặc biệt trong bảng ASCII tạo thành khung là 1 table có nhiều ô trống (tuỳ theo soTrang và soKhungTrang) để hiện thị trang.



* pageOut(int \*pages, int soTrang):sử dụng vòng lăp for xuất ra các trang là các phần tử có trong mảng pages ra màng hình.
* display(int \*tmpArr, int soKhungTrang, int soTrang, int column):sử dụng vòng lặp for để xuất ra màn hình các trang là phần tử trong mảng tmpArr có giá trị khác -1 theo hàng dọc (sử dụng hàm gotoxy(x, y)), ngược lại phần tử nào có giá trị là -1 thì sẽ in ra kí tự rỗng. .
* **fifoPageReplacement**(int \*pages, int soTrang, int soKhungTrang):
  + Khai báo 1 mảng tạm(tmpArr) có số phần tử bằng soKhungTrang.
  + Tiếp tục khai báo các biến “soLoiTrang” và “x” rồi gán cho cả hai giá trị là 0. Biến “soLoiTrang” dùng để đếm tổng số lỗi trang và là kết quả trả về của hàm, biến “x” dùng để xác định vị trí của phần tử trong mảng tạm(được dùng khi xuất hiện lỗi trang và số lỗi trang bé hơn hoặc bằng số khung trang).
  + Gán cho tất cả các phần tử trong mảng tmpArr giá trị -1 bằng cách gọi hàm init(tmpArr, soKhungTrang).
  + Gọi hàm pagesOut(pages, soTrang) để hiển thị ra màn hình danh sách các phần tử trong mảng pages.



* + Sau đó khởi tạo vòng lặp for có giá trị i chạy từ 0 đến soTrang-1(cuối mảng pages).
    - Trong vòng lặp thứ nhất, đầu tiên ta sẽ khởi tạo biến exists(bool) và gán cho nó giá trị là false.
    - Sử dụng hàm isExists để kiểm tra xem phần tử pages[i] đã tồn tại trong mảng tạm tmpArr hay chưa. Nếu đã tồn tại thì giảm giá trị biến soLoiTrang xuống 1 đơn vị và gán lại biến exists = true.
    - Tăng giá trị biến soLoiTrang lên 1 đơn vị (nếu trước đó phát hiện 1 lỗi trang thì sẽ tăng lên (-1) + 1 = 0 soLoiTrang sau 1 vòng lặp)
    - Kiểm tra điều kiện, nếu: soLoiTrang <= soKhungTrang và không phát hiện lỗi trang trong vòng lặp hiện tại thì gán pages[i] cho tmpArr[x] và tăng x lên 1 đơn vị (x sẽ có giá trị lớn nhất bằng với số khung trang(frame)). Hoặc nếu chỉ thoả mãn điều kiện exists == false mà không thoả điều kiện soLoiTrang <= soKhungTrang thì ta sẽ gán pages[i] cho tmpArr[idx], với idx là vị trí của mảng được tính theo công thức: idx = (soLoiTrang – 1) % soKhungTrang.
    - Nếu exists == false (có lỗi trang) thì ta sẽ xuất ra màn hình kí tự “\*” ở vị trí thích hợp để đánh dấu, thể hiện việc có lỗi trang đã xuất hiện ở bước này.
    - Gọi hàm drawBorder(soTrang, soKhungTrang) để tạo ra khung chứa các trang.
    - display: biểu diễn ra màn hình các phần tử của hàm tmpArr .
    - Tạo hiệu ứng Sleep(300) rồi chuyển sang vòng lặp kế tiếp.
* Kết quả trả về của hàm này sẽ là soLoiTrang.

**Một vài hình ảnh minh hoạ:**

Xét chuỗi truy xuất bộ nhớ có 15 trang: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3

Trường hợp có 3 khung trang:



Trường hợp có 4 khung trang:



Trường hợp có 5 khung trang:



 Xét chuỗi truy xuất bộ nhớ có 20 trang: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Trường hợp có 3 khung trang:

Trường hợp có 4 khung trang:



 Trường hợp có 5 khung trang: