**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---🙠**🕮**🙢---**



BÁO CÁO THỰC TẬP CƠ SỞ

**ĐỀ TÀI: MÔ PHỎNG CÁC CHIẾN LƯỢC THAY THẾ BỘ NHỚ**

|  |  |
| --- | --- |
| ***Giáo viên hướng dẫn*** | ***Nguyễn Thị Thu Thúy*** |
| ***Sinh viên thực hiện thứ nhất:*** | ***Trần Việt Thiện*** |
| ***Mã số sinh viên:*** | ***61131383*** |
| ***Sinh viên thực hiện thứ hai:*** | ***Nguyễn Phan Hữu Thọ*** |
| ***Mã số sinh viên:*** | ***61132696*** |
| ***Lớp:*** | ***61CNTT - 2*** |
|  | |

**---🙠**🕮**🙢---**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---🙠**🕮**🙢---**



BÁO CÁO THỰC TẬP CƠ SỞ

**ĐỀ TÀI: MÔ PHỎNG CÁC CHIẾN LƯỢC THAY THẾ BỘ NHỚ**

|  |  |
| --- | --- |
| ***Giáo viên hướng dẫn*** | ***Nguyễn Thị Thu Thúy*** |
| ***Sinh viên thực hiện thứ nhất:*** | ***Trần Việt Thiện*** |
| ***Mã số sinh viên:*** | ***61131383*** |
| ***Sinh viên thực hiện thứ hai:*** | ***Nguyễn Phan Hữu Thọ*** |
| ***Mã số sinh viên:*** | ***61132696*** |
| ***Lớp:*** | ***61CNTT - 2*** |
|  | |

**---🙠**🕮**🙢---**

***MỤC LỤC***

[1](#_Toc114301599)

[***MỤC LỤC*** 3](#_Toc114301600)

[**I.** **Bộ nhớ ảo là gì?** 4](#_Toc114301601)

[**II.** **Cơ chế thực hiện bộ nhớ ảo** 4](#_Toc114301602)

[**1.** **Phân trang theo yêu cầu (demand paging)** 4](#_Toc114301603)

[**2.** **Cơ chế phần cứng** 5](#_Toc114301604)

[**3.** **Lỗi trang** 5](#_Toc114301605)

[**Chương II: THAY THẾ TRANG BỘ NHỚ** 7](#_Toc114301606)

[**I.** **TỔNG QUAN** 7](#_Toc114301607)

[**II.** **CÁC THUẬT TOÁN THAY THẾ TRANG** 7](#_Toc114301608)

[**Ví dụ:** 8](#_Toc114301609)

[**Chương III: CÀI ĐẶT CHƯƠNG TRÌNH** 13](#_Toc114301610)

[**I.** **GIỚI THIỆU VỀ PHẦN MỀM DEV C** 13](#_Toc114301611)

[**II.** **HƯỚNG DẪN CÀI ĐẶT PHẦN MỀM DEV C** 14](#_Toc114301612)

[**III.** **HƯỚNG DẪN CÀI ĐẶT THƯ VIỆN GRAPHICS** 21](#_Toc114301613)

[**IV.** **CHƯƠNG TRÌNH MÔ PHỎNG** 24](#_Toc114301614)

[**1. Cấu trúc chương trình:** 24](#_Toc114301615)

[**1.1 Thư viện sử dụng** 24](#_Toc114301616)

[**1.2 Các biến quan trọng** 24](#_Toc114301617)

[**1.3** 24](#_Toc114301618)

[**Các hàm xử lý chung** 25](#_Toc114301619)

[**1.4 Chương trình chính** 28](#_Toc114301620)

[**4.** **Chương trình hoàn chỉnh:** 31](#_Toc114301621)

[**4.1** **Thuật toán FIFO (First In First Out)** 31](#_Toc114301622)

[**4.2 Thuật toán LRU (Least Recently Used)** 40](#_Toc114301623)

[**4.3 Thuật toán OPT (Optimal)** 46](#_Toc114301624)

[**4.4 Thuật toán Second Chance** 53](#_Toc114301625)

[**Chương IV: KẾT LUẬN** 59](#_Toc114301626)

[**I.** **KẾT QUẢ ĐẠT ĐƯỢC** 59](#_Toc114301627)

[**II.** **ƯU ĐIỂM** 59](#_Toc114301628)

[**III.** **HẠN CHẾ** 59](#_Toc114301629)

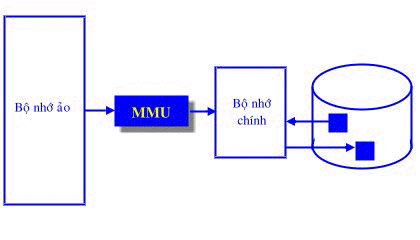
[**IV.** **HƯỚNG PHÁT TRIỂN** 59](#_Toc114301630)

[**V.** **KẾT LUẬN** 59](#_Toc114301631)

**Chương I: CƠ SỞ LÝ THUYẾT**

1. **Bộ nhớ ảo là gì?**

Bộ nhớ ảo là một kỹ thuật cho phép xử lý một tiến trình không được nạp toàn bộ vào bộ nhớ vật lý. Bộ nhớ ảo mô hình hoá bộ nhớ như một bảng lưu trữ rất lớn và đồng nhất, tách biệt hẳn khái niệm không gian địa chỉ và không gian vật lý. Người sử dụng chỉ nhìn thấy và làm việc trong không gian địa chỉ ảo, việc chuyển đổi sang không gian vật lý do hệ điều hành thực hiện với sự trợ giúp của các cơ chế phần cứng cụ thể.



1. **Cơ chế thực hiện bộ nhớ ảo**

Bộ nhớ ảo thường được thực hiện với kỹ thuật phân trang theo yêu cầu (demand paging). Cũng có thể sử dụng kỹ thuật phân đoạn theo yêu cầu (demand segmentation) để cài đặt bộ nhớ ảo, tuy nhiên việc cấp phát và thay thế các phân đoạn phức tạp hơn thao tác trên trang, vì kích thước không bằng nhau của các đoạn.

1. **Phân trang theo yêu cầu (demand paging)**

Một hệ thống phân trang theo yêu cầu là hệ thống sử dụng kỹ thuật phân trang kết hợp với kỹ thuật swapping. Một tiến trình được xem như một tập các trang, thường trú trên bộ nhớ phụ (thường là đĩa). Khi cần xử lý, tiến trình sẽ được nạp vào bộ nhớ chính. Nhưng thay vì nạp toàn bộ chương trình, chỉ những trang cần thiết trong thời điểm hiện tại mới được nạp vào bộ nhớ. Như vậy một trang chỉ được nạp vào bộ nhớ chính khi có yêu cầu.

Với mô hình này, cần cung cấp một cơ chế phần cứng giúp phân biệt các trang đang ở trong bộ nhớ chính và các trang trên đĩa. Có thể sử dụng lại bit valid-invalid nhưng với ngữ nghĩa mới:

 valid: trang tương ứng là hợp lệ và đang ở trong bộ nhớ chính.

 invalid: hoặc trang bất hợp lệ (không thuộc về không gian địa chỉ của tiến trình) hoặc trang hợp lệ nhưng đang được lưu trên bộ nhớ phụ.

Một phần tử trong bảng trang mộ tả cho một trang không nằm trong bộ nhớ chính, sẽ được đánh dấu invalid và chứa địa chỉ của trang trên bộ nhớ phụ.

1. **Cơ chế phần cứng**

Cơ chế phần cứng hỗ trợ kỹ thuật phân trang theo yêu cầu là sự kết hợp của cơ chế hỗ trợ kỹ thuật phân trang và kỹ thuật swapping:

 Bảng trang: Cấu trúc bảng trang phải cho phép phản ánh tình trạng của một trang là đang nằm trong bộ nhớ chính hay bộ nhớ phụ.

 Bộ nhớ phụ: Bộ nhớ phụ lưu trữ những trang không được nạp vào bộ nhớ chính. Bộ nhớ phụ thường được sử dụng là đĩa, và vùng không gian đĩa dùng để lưu trữ tạm các trang trong kỹ thuật swapping được gọi là không gian swapping.

A screenshot of a computer

Description automatically generated with low confidence

**Hình 1.1**Bảng trang với một số trang trên bộ nhớ phụ

1. **Lỗi trang**

Truy xuất đến một trang được đánh dấu bất hợp lệ sẽ làm phát sinh một lỗi trang (page fault). Khi dò tìm trong bảng trang để lấy các thông tin cần thiết cho việc chuyển đổi địa chỉ, nếu nhận thấy trang đang được yêu cầu truy xuất là bất hợp lệ, cơ chế phần cứng sẽ phát sinh một ngắt để báo cho hệ điều hành. Hệ điều hành sẽ xử lý lỗi trang như sau:

 Kiểm tra truy xuất đến bộ nhớ là hợp lệ hay bất hợp lệ

 Nếu truy xuất bất hợp lệ: kết thúc tiến trình

 Ngược lại: đến bước 3

 Tìm vị trí chứa trang muốn truy xuất trên đĩa.

 Tìm một khung trang trống trong bộ nhớ chính:

 Nếu tìm thấy: đến bước 5

 Nếu không còn khung trang trống, chọn một khung trang « nạn nhân » và chuyển trang « nạn nhân » ra bộ nhớ phụ (lưu nội dung của trang đang chiếm giữ khung trang này lên đĩa), cập nhật bảng trang tương ứng rồi đến bước 5

 Chuyển trang muốn truy xuất từ bộ nhớ phụ vào bộ nhớ chính : nạp trang cần truy xuất vào khung trang trống đã chọn (hay vừa mới làm trống ) ; cập nhật nội dung bảng trang, bảng khung trang tương ứng.

 Tái kích hoạt tiến trình người sử dụng.

A picture containing diagram

Description automatically generated

**Hình 1.2**Các giai đoạn xử lý lỗi trang

**Chương II: THAY THẾ TRANG BỘ NHỚ**

1. **TỔNG QUAN**

Khi xảy ra một lỗi trang, cần phải mang trang vắng mặt vào bộ nhớ. Nếu không có một khung trang nào trống, hệ điều hành cần thực hiện công việc thay thế trang – chọn một trang đang nằm trong bộ nhớ mà không được sử dụng tại thời điểm hiện tại và chuyển nó ra không gian swapping trên đĩa để giải phóng một khung trang dành chỗ nạp trang cần truy xuất vào bộ nhớ.

Như vậy nếu không có khung trang trống, thì mỗi khi xảy ra lỗi trang cần phải thực hiện hai thao tác chuyển trang: chuyển một trang ra bộ nhớ phụ và nạp một trang khác vào bộ nhớ chính. Có thể giảm bớt số lần chuyển trang bằng cách sử dụng thêm một bit cập nhật (dirty bit). Bit này được gắn với mỗi trang để phản ánh tình trạng trang có bị cập nhật hay không: giá trị của bit được cơ chế phần cứng đặt là 1 mỗi lần có một từ được ghi vào trang, để ghi nhận nội dung trang có bị sửa đổi. Khi cần thay thế một trang, nếu bit cập nhật có giá trị là 1 thì trang cần được lưu lại trên đĩa, ngược lại, nếu bit cập nhật là 0, nghĩa là trang không bị thay đổi, thì không cần lưu trữ trang trở lại đĩa.

|  |  |  |
| --- | --- | --- |
| số hiệu trang | bit valid-invalid | dirty bit |

Cấu trúc một phần tử trong bảng trang

Sự thay thế trang là cần thiết cho kỹ thuật phân trang theo yêu cầu. Nhờ cơ chế này, hệ thống có thể hoàn toàn tách rời bộ nhớ ảo và bộ nhớ vật lý, cung cấp cho lập trình viên một bộ nhớ ảo rất lớn trên một bộ nhớ vật lý có thể bé hơn rất nhiều lần.

1. **CÁC THUẬT TOÁN THAY THẾ TRANG**

Vấn đề chính khi thay thế trang là chọn lựa một trang để chuyển ra bộ nhớ phụ. Có nhiều thuật toán thay thế trang khác nhau, nhưng tất cả cùng chung một mục tiêu: chọn trang là trang mà sau khi thay thế sẽ gây ra ít lỗi trang nhất.

Có thể đánh giá hiệu qủa của một thuật toán bằng cách xử lý trên một chuỗi các địa chỉ cần truy xuất và tính toán số lượng lỗi trang phát sinh.

Ví dụ: Giả sữ theo vết xử lý của một tiến trình và nhận thấy tiến trình thực hiện truy xuất các địa chỉ theo thứ tự sau:

0100, 0432, 0101, 0162, 0102, 0103, 0104, 0101, 0611, 0102, 0103,0104, 0101, 0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105

Nếu có kích thước của một trang là 100 bytes, có thể viết lại chuỗi truy xuất trên giản lược hơn như sau:

1, 4, 1, 6, 1, 6, 1, 6, 1

Để xác định số các lỗi trang xảy ra khi sử dụng một thuật toán thay thế trang nào đó trên một chuỗi truy xuất cụ thể, còn cần phải biết số lượng khung trang sử dụng trong hệ thống.

**Ví dụ:**

Cho chuỗi truy xuất 2, 3, 2, 1, 5, 2, 4 và số khung trang là 3. Sử dụng các thuật toán FIFO, LRU, OTP, Second Chance để biểu diễn quá trình thay thế trang, đánh dấu “\*” vào những trang có lỗi và in ra tổng số lỗi trang.

* **Thuật toán FIFO (First In First Out)**
  + Trang nào vào trước thì bị đổi ra trước.
  + Là thuật toán đơn giản nhất.
  + Trang bị trao đổi là trang nằm lâu nhất trong bộ nhớ.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | | 3 | | 2 | | 1 | | 5 | | 2 | | 4 | |
| 2 | 2 | | 2 | | 2 | | 5 | | 5 | | 5 | |
|  | 3 | | 3 | | 3 | | 3 | | 2 | | 3 | |
|  |  | |  | | 1 | | 1 | | 1 | | 4 | |
| \* | \* | | \* | |  | | \* | | \* | | \* | |

Tổng số lỗi trang: 6

* Ban đầu, 2 ô trống nên phần tử 2, 3, đầu tiên được phân bổ vào các ô trống. Lỗi trang = 2
* Tiếp theo còn một ô trống, 2 vào nhưng đã có sẵn trong trang. Lỗi trang = 0
* Tiếp theo còn một ô trống, 1 không có sẵn trong trang nên được phân vào ô trống. Lỗi trang = 1
* Bây giờ phần tử 5 xuất hiện. Nó không hiện diện trong bộ nhớ vì vậy trang 2 là phần tử cũ nhất sẽ được thay thế. Lỗi trang = 1
* Sau phần tử 5, phần tử 2 xuất hiện. Phần tử này cũng không có sẵn trong bộ nhớ vì vậy nó được thay thế bằng vị trí trang cũ nhất là 3. Lỗi trang = 1
* Cuối cùng, phần tử 4 xuất hiện, không có sẵn trong bộ nhớ nên nó thay thế phần tử 1 cũ nhất. Lỗi trang = 1
* **Thuật toán LRU (Least Recently Used)**
  + Trang bị đổi là trang mà thời gian từ lần truy cập cuối cùng đến thời điểm hiện tại là lâu nhất.
  + Theo nguyên tắc cục bộ về thời gian, đó là trang ít có khả năng sử dụng nhất trong tương lai.
  + Thực tế LRU cho kết quả tốt gần như phương pháp đổi trang tối ưu.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 2 | 1 | 5 | 2 | 4 |
| 2 | 2 | 2 | 2 | 2 | 5 | 2 |
|  | 3 | 3 | 3 | 5 | 2 | 5 |
|  |  |  | 1 | 1 | 1 | 4 |
| \* | \* |  | \* | \* |  | \* |

Tổng số lỗi trang: 5

* Ban đầu, 2 ô trống nên phần tử 2, 3, đầu tiên được phân bổ vào các ô trống. Lỗi trang = 2
* Tiếp theo còn một ô trống, 2 vào nhưng đã có sẵn trong trang. Lỗi trang = 0
* Tiếp theo còn một ô trống, 1 không có sẵn trong trang nên được phân vào ô trống. Lỗi trang = 1
* Bây giờ phần tử 5 xuất hiện. Nó không hiện diện trong bộ nhớ vì vậy nó sẽ thay thế phần tử 3 có lần truy cập cuối cùng đến thời điểm hiện tại là lâu nhất. Lỗi trang = 1
* Sau khi 5 phần tử 2 xuất hiện. Phần tử này có sẵn trong bộ nhớ. Lỗi trang = 0
* Cuối cùng, phần tử 4 xuất hiện, không có sẵn trong bộ nhớ nên nó thay thế phần tử 1 có lần truy cập cuối cùng đến thời điểm hiện tại là lâu nhất. Lỗi trang = 1
* **Thuật toán OPT (Optimal)**
  + Chọn trang sẽ không được dùng tới trong khoảng thời gian lâu nhất để đổi.
  + Cho phép giảm tối thiểu sự kiện thiếu trang và do đó là tối ưu theo tiêu chuẩn này.
  + Hệ điều hành không đoán trước được nhu cầu sử dụng các trang trong tương lai.

Không áp dụng trong thực tế mà chỉ để so sánh với các chiến lược khác.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 2 | 1 | 5 | 2 | 4 |
| 2 | 2 | 2 | 2 | 2 | 2 | 4 |
|  | 3 | 3 | 3 | 5 | 5 | 5 |
|  |  |  | 1 | 1 | 1 | 1 |
| \* | \* |  | \* | \* |  | \* |

Tổng số lỗi trang: 5

* Ban đầu, 2 ô trống nên phần tử 2, 3, đầu tiên được phân bổ vào các ô trống. Lỗi trang = 2
* Tiếp theo còn một ô trống, 2 vào nhưng đã có sẵn trong trang. Lỗi trang = 0
* Tiếp theo còn một ô trống, 1 không có sẵn trong trang nên được phân vào ô trống. Lỗi trang = 1
* Bây giờ phần tử 5 xuất hiện. Nó không hiện diện trong bộ nhớ. Xem xét vị trí thay thế là các phần tử 2,3,1 và nhìn về bên phải của chuỗi truy xuất, ta thấy phần tử 3 và 1 sẽ là các trang sẽ không được dùng tới trong khoảng thời gian lâu nhất, vì vậy phần tử 5 sẽ thay thế một trong hai phần tử 3 . Lỗi trang = 1
* Sau khi 5 phần tử 2 xuất hiện. 2 có sẵn trong bộ nhớ. Lỗi trang = 0
* Cuối cùng, phần tử 4 xuất hiện. Nó không hiện diện trong bộ nhớ. Xem xét vị trí thay thế là các phần tử 2,5,1 và nhìn về bên phải của chuỗi truy xuất, ta thấy phần tử 2, 5 và 1 sẽ là các trang sẽ không được dùng tới trong khoảng thời gian lâu nhất, vì vậy phần tử 5 sẽ thay thế một trong 3 phần tử, và trang bị thay thế là 2 . Lỗi trang = 1
* **Thuật toán Second Chance** 
  + Bản chất là FIFO
  + Cần có bit tham chiếu. Nếu bit tham chiếu =0 thì thay trang. Trái lại cho trang đó cơ hội thứ hai và chuyển đến trang tiếp sau.
* Thiết lập bit tham chiếu = 0.
* Để trang lại trong bộ nhớ.
* Thay trang kế tiếp (theo FIFO) với luật tương tự.
  + Một cách thực hiện giải thuật là sử dụng queue vòng tròn.

0,1 là các bit tham chiếu.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 |  | 3 |  | 2 |  | 1 |  | 5 |  | 2 |  | 4 |
| 0 | 2 | 0 | 2 | 1 | 2 | 1 | 2 | 0 | 2 | 1 | 2 | 0 | 2 |
|  |  | 0 | 3 | 0 | 3 | 0 | 3 | 0 | 5 | 0 | 5 | 0 | 5 |
|  |  |  |  |  |  | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 4 |
|  | \* |  | \* |  |  |  | \* |  | \* |  |  |  | \* |

Tổng số lỗi trang: 5

* Đầu tiên 2 đi vào, thay vào ô trống đầu tiên, bit tham chiếu của 2 được đặt bằng 0. Lỗi trang =1
* Tiếp theo 3 đi vào, ô trống thứ nhất đã có 2 nên thay 3 vào ô trống thứ 2, bit tham chiếu của 3 được đặt bằng 0. Lỗi trang = 2.
* Bây giờ lại có 2 đi vào, 2 đã có trong bộ nhớ, nhưng vì 2 đã được tham chiếu lại nên bit tham chiếu của 2 sẽ được đặt lại bằng 1. Lỗi trang = 2.
* Tiếp theo 1 đi vào, thêm 1 vào ô trống thứ 3, bit tham chiếu của 1 được đặt bằng 0. Lỗi trang = 3.
* Tiếp theo 5 đi vào, 5 không có trong bộ nhớ và khung trang cũng đã hết chỗ trống vậy trang nào sẽ được thay ra? Vì second chance là biến thể của FIFO nên thường sẽ áp dụng quy tắc (trang nào vào trước sẽ được thay ra trước) nên 5 sẽ thay vào chỗ của 2, nhưng vì đây là thuật toán second chance nên cần xem xét thêm một yếu tố nữa đó là giá trị bit tham chiếu. Ở đây trang nạn nhân là 2 theo FIFO nhưng bit tham chiếu của 2 là 1 nên nó có cơ hội thứ 2 và trong trường hợp này nạn nhân tiếp theo tức là 3(là trang cũ nhất) sẽ là trang được 5 thay thế. Bit tham chiếu của 2,5,1 đều sẽ được đặt lại bằng 0. Lỗi trang = 4.
* Bây giờ lại có 2 đi vào, 2 đã có trong bộ nhớ, nhưng vì 2 đã được tham chiếu lại nên bit tham chiếu của 2 sẽ được đặt lại bằng 1. Lỗi trang = 4.
* Cuỗi cùng là 4 đi vào, dựa theo quy tắc second chance nên 4 sẽ thay cho 1, Bit tham chiếu của 2,5,4 đều sẽ được đặt lại bằng 0. Lỗi trang = 5.

**Chương III: CÀI ĐẶT CHƯƠNG TRÌNH**

1. **GIỚI THIỆU VỀ PHẦN MỀM DEV C**

Dev-C++ là một môi trường phát triển tích hợp tự do (IDE) được phân phối dưới hình thức giấy phép Công cộng GNU hỗ trợ việc lập trình bằng C/C++. Nó cũng nằm trong bộ trình dịch mã nguồn mở MinGW. Chương trình IDE này được viết bằng ngôn ngữ Delphin.

Dự án phát triển Dev-C++ được lưu trữ trên SourceForge. Dev-C++ nguyên được phát triển bởi một lập trình viên có tên là Colin Laplace và chỉ chạy trên hệ điều hành Microsoft Windows.

Bloodshed Dev-C++ là IDE dành cho C/C++. C/C++ là một ngôn ngữ lập trình cực mạnh, có khả năng tương tác cao và thích ứng với các hệ điều hành khác nhau. Hiện nay, có rất nhiều IDE hỗ trợ lập trình C/C++ như Turbo C, Visual C++… và Bloodshed Dev-C++ là IDE Portable đầu tiên.

Graphical user interface, text, application, email

Description automatically generated

Chương trình Dev-C++ hỗ trợ đầy đủ các chức năng mà một IDE chuyên nghiệp, từ việc soạn thảo, trợ giúp đến debug và hỗ trợ thư viện cho các lập trình viên C/C++ (kể cả viết chương trình cho DOS và chương trình cho Windows). Với IDE này, bạn dễ dàng tạo ra một chương trình C/C++ một cách chuyên nghiệp, nhanh chóng và đơn giản.

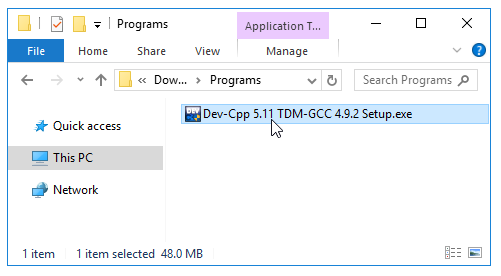
1. **HƯỚNG DẪN CÀI ĐẶT PHẦN MỀM DEV C**

Tải Dev C++ tại: <https://sourceforge.net/projects/orwelldevcpp/>

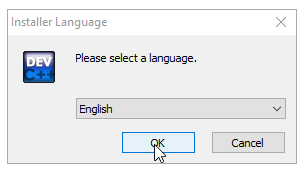
Graphical user interface, text

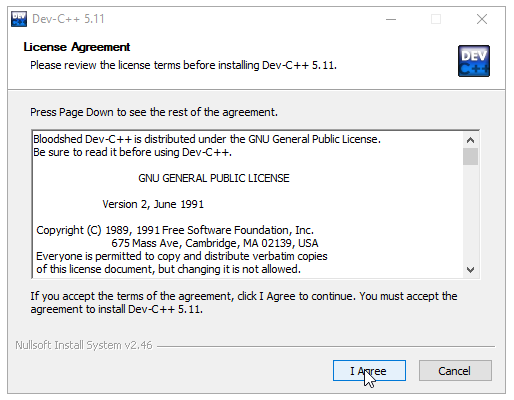
Description automatically generated

Click đúp vào file .exe vừa tải về



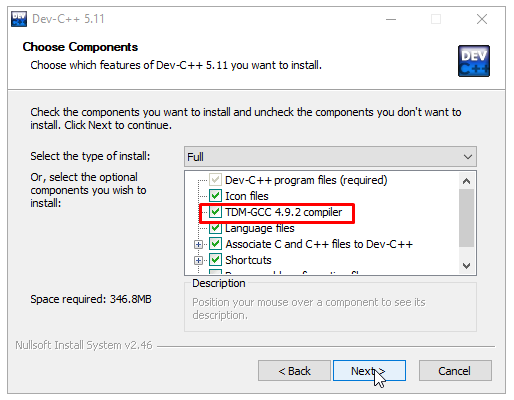
Chọn ngôn ngữ cài đặt

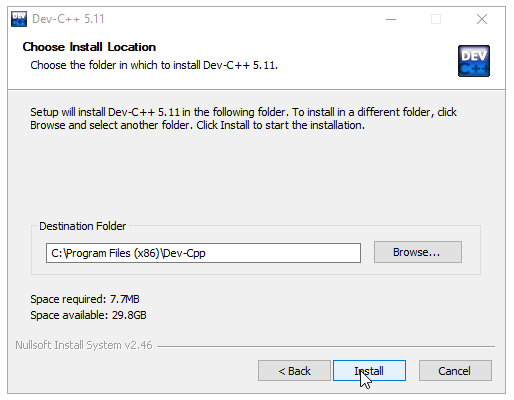


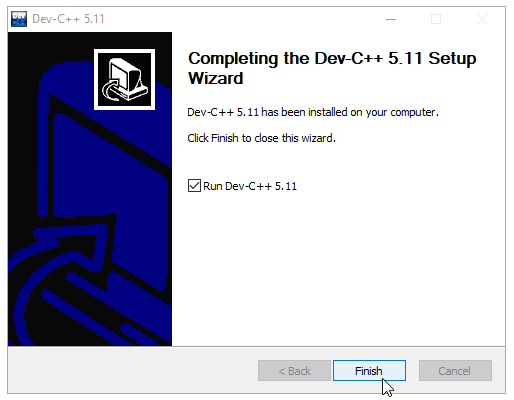


Bộ cài Dev C++ tích hợp sẵn trình biên dịch ngôn ngữ C/C++ là Boost C++, nếu máy bạn đã cài sẵn trình compiler C/C++ như MinGW hay Cygwin thì có thể bỏ chọn ô check box ở dưới. (Mình khuyên cài MinGW vì bản compiler đi sẵn với Dev C++ khá là cũ)

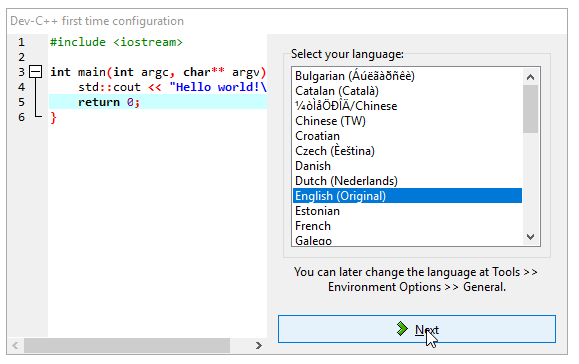
([Hướng dẫn cài đặt trình biên dịch C/C++ MinGW (gcc)](https://codecute.com/c/huong-dan-cai-dat-trinh-bien-dich-c-c-mingw-gcc.html))

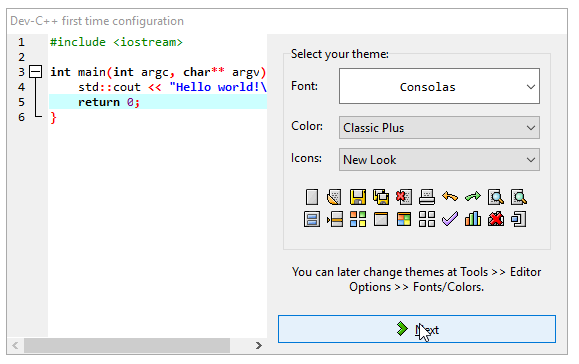


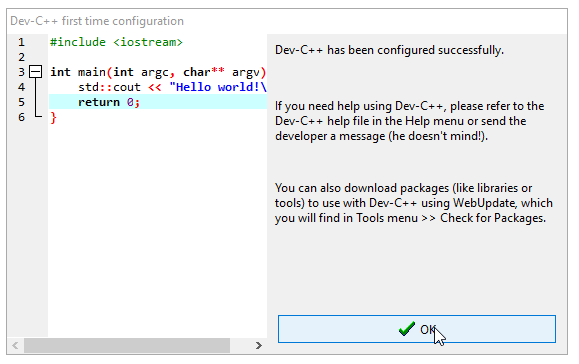


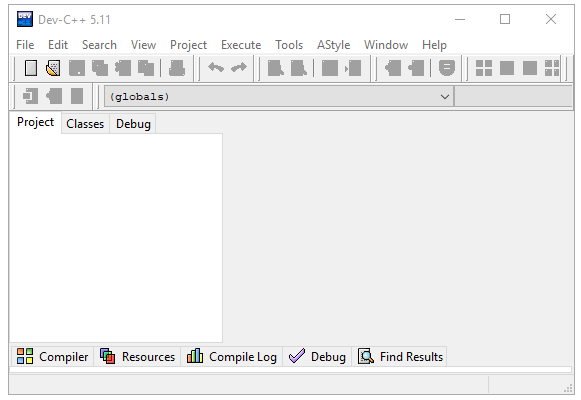


Lần đầu chạy Dev C++ bạn sẽ cần cấu hình một số thông tin như ngôn ngữ, giao diện (font, color)







Đây là giao diện của Dev C++  


**Ví dụ C/C++ Hello World trên Dev C++**

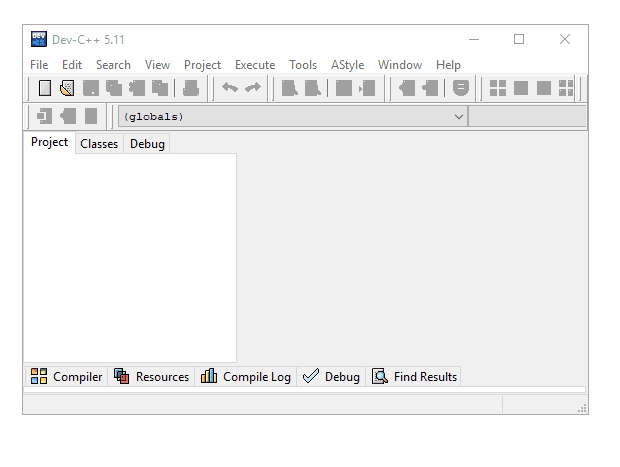
Chọn File/New/Source File (Hoặc ấn Ctrl+N) để tạo file

Copy nội dung sau vào file vừa tạo:

* #include <stdio.h>
* int main() {
* printf("Hello World!");
* return 0;
* }

Sau đó lưu lại thành file **Hello.cpp** (với C++) hoặc **Hello.c** (với C)

Trong ví dụ này mình lưu thành file **Hello.cpp** và lưu ở folder **D:\programming**



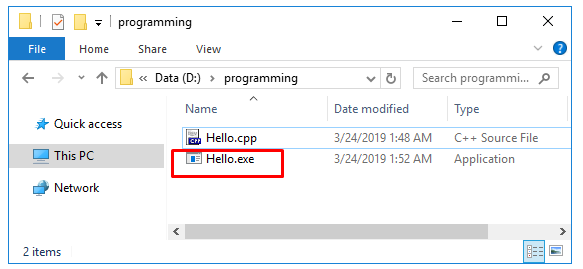
Chạy file **Hello.cpp** trên bằng cách chọn ***Execute/ Compile & Run*** (hoặc ấn F11) (Compile là để biên dịch thành file Hello.exe, Run là để chạy file .exe vừa biên dịch) (Lưu ý, nếu bạn không cài trình biên dịch C/C++ (**Boost C++**) tích hợp sẵn với với Dev C++ thì bạn phải [cấu hình compiler cho Dev C++](https://codecute.com/c/huong-dan-cau-hinh-compiler-mingw-cygwin-cho-dev-c.html))

Kết quả:

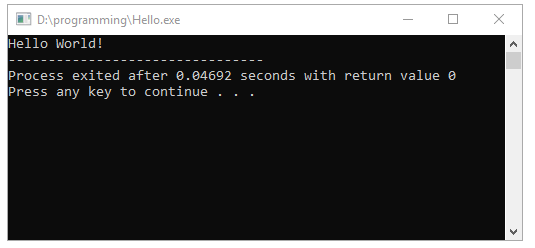
File **Hello.exe** được tạo ra

[Graphical user interface, text, application

Description automatically generated](https://codecute.com/wp-content/uploads/2019/03/dev-hello-2.gif)

[](https://codecute.com/wp-content/uploads/2019/03/dev-c-19.png)

Màn hình hiển thị dòng chữ ‘**Hello World!**‘

[](https://codecute.com/wp-content/uploads/2019/03/dev-c-18.png)

1. **HƯỚNG DẪN CÀI ĐẶT THƯ VIỆN GRAPHICS**

Bước 1: Link tải thư viện: [Tại đây](https://renewdock755.weebly.com/graphicsh-file-for-dev-c-download.html)

Graphical user interface, application

Description automatically generated

Bước 2: Add các tập tin trong thư viện graphics vào devC++

Copy 2 file "6-ConsoleAppGraphics.template" và "ConsoleApp\_cpp\_graph.txt" vào thư mục "(Path)\Templates".

Graphical user interface

Description automatically generated with medium confidence

Copy 2 file "graphics.h" và "winbgim.h" vào thư mục "(Path)\MinGW64\x86\_64-w64-mingw32\include".

Ảnh có chứa bàn

Mô tả được tạo tự động

Copy file "libbgi.a" vào thư mục "(Path)\MinGW64\x86\_64-w64-mingw32\lib".

Graphical user interface, application

Description automatically generated

Bước 3: Tạo project

Khởi động DevC++ , chọn file->new->project

**Graphical user interface, application

Description automatically generated**

Chọn vị trí lưu và nhấn Save

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. **CHƯƠNG TRÌNH MÔ PHỎNG**

**1. Cấu trúc chương trình:**

**1.1 Thư viện sử dụng**



<bits / stdc ++. h> là một tệp tiêu đề. Tệp này bao gồm tất cả thư viện tiêu chuẩn.

Trong cách tiếp cận kỹ thuật phần mềm, chúng ta nên giảm thiểu việc khai báo thư viện. Sử dụng tệp tiêu đề này, nó sẽ khai báo tất cả các thư viện. Vì vậy, nó có thể làm tăng thời gian biên dịch và kích thước chương trình. Một số nhược điểm lớn của tệp tiêu đề này:

<bits.stc ++. h> này không phải là tệp tiêu đề chuẩn của thư viện GNU C ++. Vì vậy, một số trình biên dịch có thể không biên dịch được mã nguồn với tệp tiêu đề này.

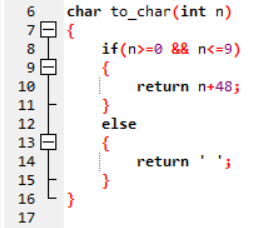
Sử dụng điều này, nó có thể yêu cầu thời gian biên dịch lâu hơn không cần thiết.

**1.2 Các biến quan trọng**

|  |  |  |
| --- | --- | --- |
| Tên biến | Kiểu giá trị | Chức năng |
| faults | Integer (kiểu số nguyên) | Đếm số lỗi trang |
| np | Integer (kiểu số nguyên) | Số lượng phần tử của chuỗi truy xuất |
| nf | Integer (kiểu số nguyên) | Số lượng khung trang |
| frame[] | Array (mảng số nguyên) | Mảng khung trang |
| page[] | Array (mảng số nguyên) | Mảng chuỗi truy xuất |
| pos | Integer (kiểu số nguyên) | Lưu vị trí |

**1.3**

**Các hàm xử lý chung**



**Hàm to\_char**: dùng để chuyển đổi kiểu số nguyên sang kiểu ký tự.

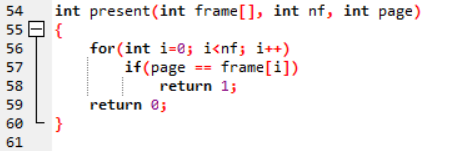
Text

Description automatically generated

**Hàm pf**: dùng để in ra tổng số lỗi trang trong màn hình đồ họa

A screenshot of a computer

Description automatically generated with low confidence

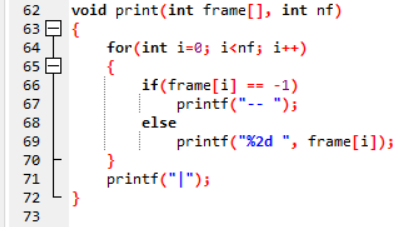


**Hàm present**

Tham số truyền vào: mảng frame[], số lượng khung trang nf, page - giá trị của trang cần thay.

Hàm kiểm tra trả về giá trị hai trường hợp:

* return 1 khi trang cần thay đã có sẵn trong khung.
* return 0 khi trang cần thay chưa có trong khung và khung còn trống.

Tham số truyền vào: mảng số nguyên frame[], số lượng khung trang nf

Duyệt từ đầu đến hết khung trang

Nếu frame[i] == -1 tức là khung trang chưa chứa giá trị nào thì đánh dấu là rỗng bằng cách in ra “-- ".

Ngược lại nếu trang có giá trị thì in giá trị của trang đó ra bằng câu lệnh printf("%2d ", frame[i]);

A picture containing text

Description automatically generated

**Hàm draw**

Hàm thực hiện mô phỏng thuật toán:

* In tiêu đề.
* In ra các khung trang và giá trị tương ứng.
* Khoanh tròn vào những ô có thay đổi giá trị.
* Đánh giấu “ \* ” vào các khung xuất hiện lỗi trang
* In ra tổng số lỗi trang.

Hàm sử dụng một số hàm có sẵn của thư viện graphics.h:

* Hàm in chuỗi ký tự:



Gồm có 2 biến tọa độ x, y để xác định vị trí in và một biến để điền giá trị của chuỗi ký tự cần in.

Vì chương trình chủ yếu sử dụng kiểu giá trị int mà hàm outextxy chỉ in được kiểu ký tự nên phải dùng đền hàm to\_char ở trên để chuyển đổi các giá trị kiểu int sang kiểu char.

* Hàm vẽ hình chữ nhật:



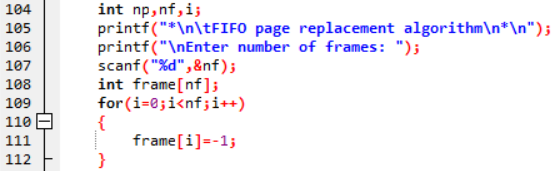
* Hàm thiết lập phông chữ:



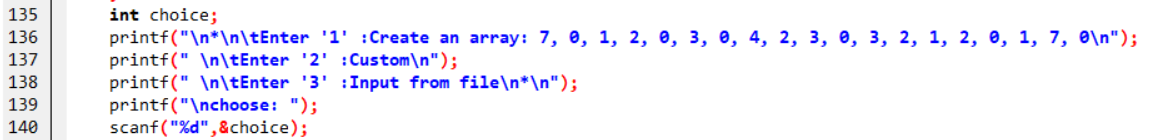
* Hàm thiết lập màu:



### **1.4 Chương trình chính**



Khai báo các biến np,nf,i.  
Hiển thị tên chương trình.  
Nhập số khung trang, những khung trang nào mà trống sẽ được gán bằng -1.

  
Khai báo 1 biến lựa chọn (choice), hiển thị 3 lựa chọn cho người dùng:

1. Tạo tự động 1 mảng gồm các phần tử: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0.
2. Tạo theo ý muốn.
3. Nhập từ file cá nhân.

Sử dụng cấu trúc rẽ nhánh để chương trình xử lý các lựa chọn của người dùng.

Text

Description automatically generated

Text

Description automatically generated

**Đọc ghi file**

Ngoài việc sử dụng thư viện chuẩn nhập xuất file được kế thừa từ ngôn ngữ C thì trong C++ có thêm 3 class phục vụ cho việc nhập xuất file là ifstream ,ofstream và fstream.

Trong đó: ifstream được sử dụng để mở file chỉ với mục đích đọc.

Dưới đây là cú pháp chuẩn cho hàm open(), là một thành viên của các đối tượng fstream, ifstream và ofstream trong C++:

Chart

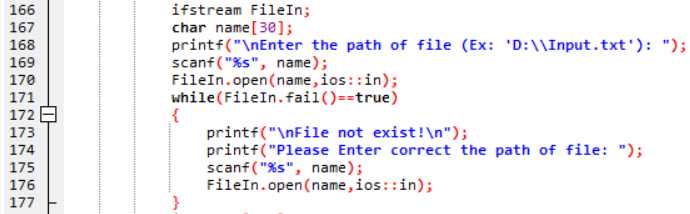
Description automatically generated

Tại đây, tham số đầu tiên xác định tên và vị trí của file để được mở và tham số thứ hai của hàm thành viên **open()** định nghĩa chế độ mà file nên được mở.

Table

Description automatically generated

**Sử dụng đọc ghi file**



char name[30]: Khai báo mảng kí tự tên là name gồm 30 phần tử để lưu đường dẫn của file chứa chuỗi truy xuất người dùng muốn nhập vào.

Dòng 104 + 105 Nhập đường dẫn của flie chứa chuỗi truy xuất mà người dùng muốn nhập vào.

FileIn.open(name,ios::in); - Mở file để đọc.

Sử dụng vòng lặp while kiểm tra xem đường dẫn file người dùng nhập vào có tồn tại hay không, nếu không vòng lặp sẽ thông báo cho người dùng “file not exist!” và bắt người dùng nhập lại đúng đường dẫn của file.

Nếu đúng đường dẫn

* khai báo mảng page gồm 100 phần tử.
* Khai báo biến np.
* Đọc chiều dài chuỗi truy xuất trong file.

Graphical user interface

Description automatically generated with low confidence

* Dùng vòng lặp for lấy từng giá trị trong file ghi vào mảng page[i].
* Cuối cùng dùng hàm add để thực hiện mô phỏng.

1. **Chương trình hoàn chỉnh:**
   1. **Thuật toán FIFO (First In First Out)**

* **Code:**

#include<graphics.h>

#include<bits/stdc++.h>

using namespace std;

int faults=0;

char to\_char(int n)

{

if(n>=0 && n<=9)

{

return n+48;

}

else

{

return ' ';

}

}

void pf(int x, int y, int n)

{

int f = faults;

for(int i=0;f>0;i++)

{

int tmp = f%10;

char a[10];

a[0] = to\_char(tmp);

settextstyle(8,0,3);

outtextxy(350-i\*15,130+43\*n+40,a);

f=f/10;

}

}

void draw(int x, int y, int frame[], int n)

{

settextstyle(8,0,4);

outtextxy(60,20,"FIFO page replacement algorithm");

setcolor(GREEN);

settextstyle(8,0,3);

char a[100];

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

{

a[i]=to\_char(frame[i]);

}

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

{

char p[10];

p[0] = a[i];

rectangle(y,x+i\*43,y+43,x+(1+i)\*43);

if(i==0)

{

outtextxy(y+15,x+12,p);

}

else

{

outtextxy(y+15,x+12+i\*43,p);

}

}

settextstyle(8,0,3);

outtextxy(35,130+43\*n+40,"Number of page faults: ");

pf(x,y,n);

}

int present(int frame[], int nf, int page)

{

for(int i=0; i<nf; i++)

if(page == frame[i])

return 1;

return 0;

}

void print(int frame[], int nf)

{

for(int i=0; i<nf; i++)

{

if(frame[i] == -1)

printf("-- ");

else

printf("%2d ", frame[i]);

}

printf("|");

}

void add(int frame[], int m, int n, int page[])

{

int pos=0;

char a[100];

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

{

a[i]=to\_char(page[i]);

char p[10];

p[0] = a[i];

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,70,p);

printf(" %2d | ",page[i]);

if(!present(frame,m,page[i]))

{

frame[pos] = page[i];

setcolor(YELLOW);

circle(52+i\*80,132+pos\*43,19);

pos = (pos+1)%m;

print(frame,m);

printf(" \*\n");

faults++;

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,130+43\*m,"\*");

draw(110,30+i\*80,frame,m);

continue;

}

print(frame,m);

printf("\n");

draw(110,30+i\*80,frame,m);

}

}

int main()

{

int np,nf,i;

printf("\*\n\tFIFO page replacement algorithm\n\*\n");

printf("\nEnter number of frames: ");

scanf("%d",&nf);

int frame[nf];

for(i=0;i<nf;i++)

{

frame[i]=-1;

}

int choice;

printf("\n\*\n\tEnter '1' :Create an array: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0\n");

printf(" \n\tEnter '2' :Custom\n");

printf(" \n\tEnter '3' :Input from file\n\*\n");

printf("\nchoose: ");

scanf("%d",&choice);

switch (choice)

{

case 1:

{

int np = 20;

int page[np]={7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0};

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

case 2:

{

printf("Enter number of pages: ");

scanf("%d",&np);

int page[np];

printf("\nEnter page value:\n");

for(i=0;i<np;i++)

{

printf("page[%d] = ",i);

scanf("%d",&page[i]);

}

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

case 3:

{

ifstream FileIn;

char name[30];

printf("\nEnter the path of file (Ex: 'D:\\Input.txt'): ");

scanf("%s", name);

FileIn.open(name,ios::in);

while(FileIn.fail()==true)

{

printf("\nFile not exist!\n");

printf("Please Enter correct the path of file: ");

scanf("%s", name);

FileIn.open(name,ios::in);

}

int page[100];

int np;

FileIn >> np;

for(i=0;i<np;i++)

{

FileIn >> page[i];

}

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

}

printf("\nNumber of page faults : %d\n\n", faults);

while(!kbhit()) delay(1); // pause screen

return 0;

}

* **Kết quả:**

Nhập số lượng khung trang

Text

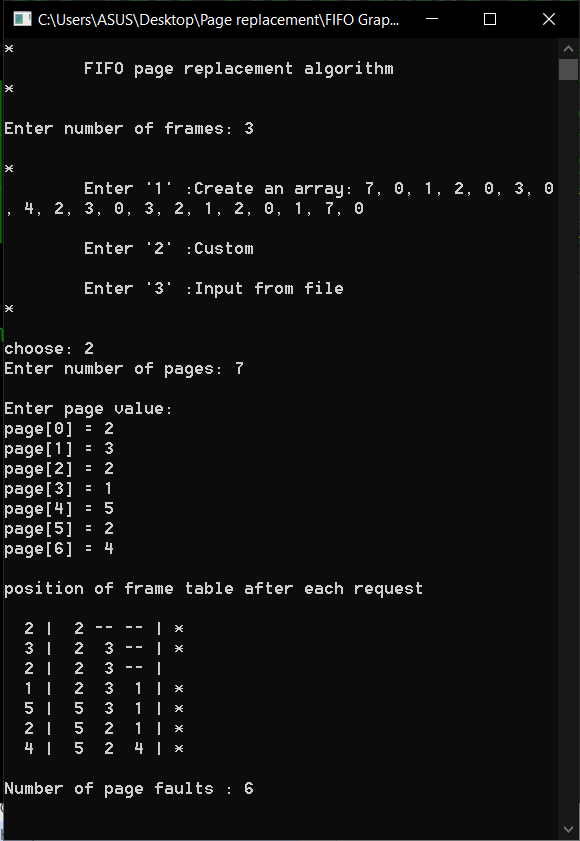
Description automatically generated

Hiển thị các lựa chọn

A screenshot of a computer

Description automatically generated with medium confidence

Chọn lựa chọn 2 và nhập chuỗi truy xuất, kết quả màn hình Console



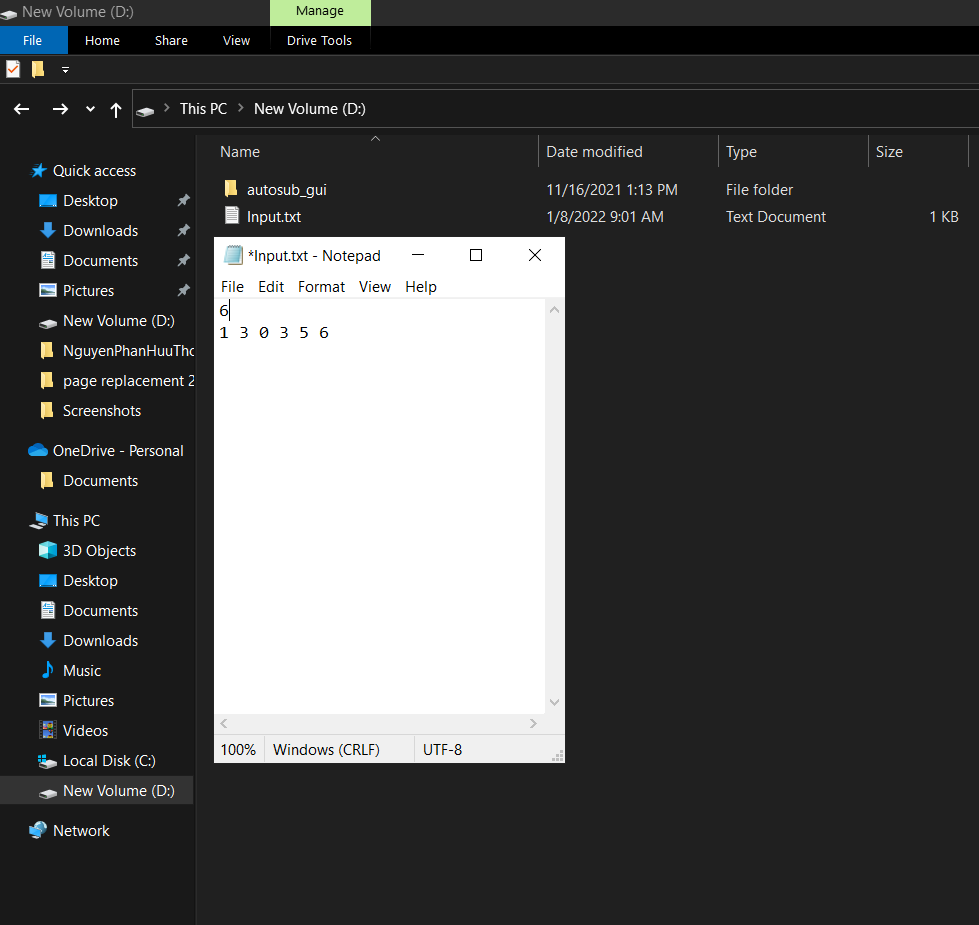
Kết quả mô phỏng bằng thư viện graphics.h

Graphical user interface

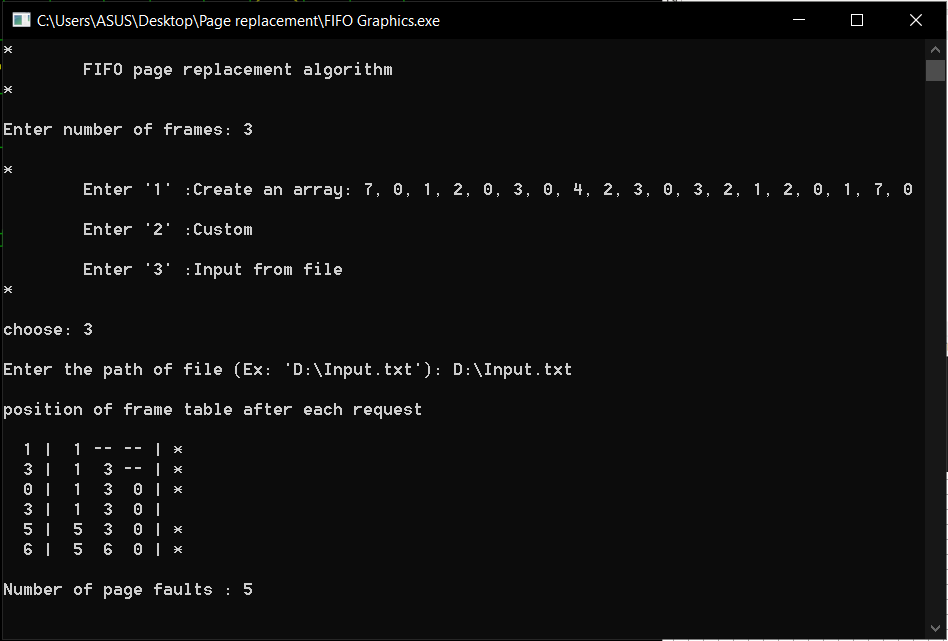
Description automatically generated

Chọn lựa chọn 3

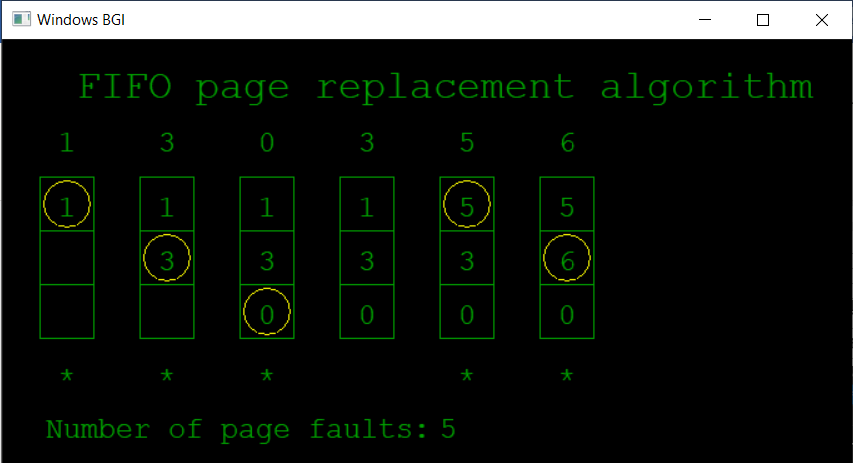
Tạo sẵn File input.txt nằm trong ổ D.



Nhập đường dẫn file và kết quả



Kết quả mô phỏng bằng thư viện graphics.h



**4.2** **Thuật toán LRU (Least Recently Used)**

* **Code:**

#include<graphics.h>

#include<bits/stdc++.h>

using namespace std;

int faults=0;

char to\_char(int n)

{

if(n>=0 && n<=9)

{

return n+48;

}

else

{

return ' ';

}

}

void pf(int x, int y, int n)

{

int f = faults;

for(int i=0;f>0;i++)

{

int tmp = f%10;

char a[10];

a[0] = to\_char(tmp);

settextstyle(8,0,3);

outtextxy(350-i\*15,130+43\*n+40,a);

f=f/10;

}

}

void draw(int x, int y, int frame[], int n)

{

settextstyle(8,0,5);

outtextxy(60,20,"LRU page replacement algorithm");

setcolor(GREEN);

settextstyle(8,0,3);

char a[100];

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

{

a[i]=to\_char(frame[i]);

}

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

{

char p[10];

p[0] = a[i];

rectangle(y,x+i\*43,y+43,x+(1+i)\*43);

if(i==0)

{

outtextxy(y+15,x+12,p);

}

else

{

outtextxy(y+15,x+12+i\*43,p);

}

}

settextstyle(8,0,3);

outtextxy(35,130+43\*n+40,"Number of page faults: ");

pf(x,y,n);

}

int present(int frame[], int nf, int page)

{

for(int i=0; i<nf; i++)

if(page == frame[i])

return 1;

return 0;

}

void print(int frame[], int nf)

{

for(int i=0; i<nf; i++)

{

if(frame[i] == -1)

printf("-- ");

else

printf("%2d ", frame[i]);

}

printf("|");

}

int findpos(int frame[], int m, int pages[], int cur, int n)

{

for(int i=0; i<m; i++)

if(frame[i] == -1)

return i;

int pos[m] = {0};

for(int i=0; i<m; i++)

{

pos[i] = -1;

for(int j=cur-1; j>=0; j--)

if(pages[j] == frame[i])

{

pos[i] = j;

break;

}

}

int min = 1000000, Pos = -1;

for(int i=0; i<m; i++)

if(min > pos[i])

{

min = pos[i];

Pos = i;

}

return Pos;

}

void add(int frame[], int m, int n, int page[])

{

char a[100];

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

{

a[i]=to\_char(page[i]);

char p[10];

p[0] = a[i];

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,70,p);

printf(" %2d | ",page[i]);

if(!present(frame,m,page[i]))

{

int pos = findpos(frame,m,page,i,n);

frame[pos] = page[i];

setcolor(YELLOW);

circle(52+i\*80,132+pos\*43,19);

print(frame,m);

printf(" \*\n");

faults++;

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,130+43\*m,"\*");

draw(110,30+i\*80,frame,m);

continue;

}

print(frame,m);

printf("\n");

draw(110,30+i\*80,frame,m);

}

}

int main()

{

int np,nf,i;

printf("\*\n\tLRU page replacement algorithm\n\*\n");

printf("\nEnter number of frames: ");

scanf("%d",&nf);

int frame[nf];

for(i=0;i<nf;i++)

{

frame[i]=-1;

}

int choice;

printf("\n\*\n\tEnter '1' :Create an array: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1\n");

printf(" \n\tEnter '2' :Custom\n");

printf(" \n\tEnter '3' :Input from file\n\*\n");

printf("\nchoose: ");

scanf("%d",&choice);

switch (choice)

{

case 1:

{

int np = 20;

int page[np]={7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1};

printf("Number of pages: 12\n");

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

case 2:

{

printf("Enter number of pages: ");

scanf("%d",&np);

int page[np];

printf("\nEnter page value:\n");

for(i=0;i<np;i++)

{

printf("page[%d] = ",i);

scanf("%d",&page[i]);

}

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

case 3:

{

ifstream FileIn;

char name[30];

printf("\nEnter the path of file (Ex: 'D:\\Input.txt'): ");

scanf("%s", name);

FileIn.open(name,ios::in);

while(FileIn.fail()==true)

{

printf("\nFile not exist!\n");

printf("Please Enter correct the path of file: ");

scanf("%s", name);

FileIn.open(name,ios::in);

}

int page[100];

int np;

FileIn >> np;

for(i=0;i<np;i++)

{

FileIn >> page[i];

}

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

}

printf("\nNumber of page faults : %d\n\n", faults);

while(!kbhit()) delay(1); // pause screen

return 0;

}

* **Kết quả:**

Chọn chọn 2 và nhập chuỗi truy xuất, kết quả màn hình console

Text

Description automatically generated

Kết quả mô phỏng bằng thư viện graphics.h

Graphical user interface, application

Description automatically generated

**4.3 Thuật toán OPT (Optimal)**

* **Code:**

#include<graphics.h>

#include<bits/stdc++.h>

using namespace std;

int faults=0;

char to\_char(int n)

{

if(n>=0 && n<=9)

{

return n+48;

}

else

{

return ' ';

}

}

void pf(int x, int y, int n)

{

int f = faults;

for(int i=0;f>0;i++)

{

int tmp = f%10;

char a[10];

a[0] = to\_char(tmp);

settextstyle(8,0,3);

outtextxy(350-i\*15,130+43\*n+40,a);

f=f/10;

}

}

void draw(int x, int y, int frame[], int n)

{

settextstyle(8,0,5);

outtextxy(60,20,"OPT page replacement algorithm");

setcolor(GREEN);

settextstyle(8,0,3);

char a[100];

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

{

a[i]=to\_char(frame[i]);

}

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

{

char p[10];

p[0] = a[i];

rectangle(y,x+i\*43,y+43,x+(1+i)\*43);

if(i==0)

{

outtextxy(y+15,x+12,p);

}

else

{

outtextxy(y+15,x+12+i\*43,p);

}

}

settextstyle(8,0,3);

outtextxy(35,130+43\*n+40,"Number of page faults: ");

pf(x,y,n);

}

int present(int frame[], int nf, int page)

{

for(int i=0; i<nf; i++)

if(page == frame[i])

return 1;

return 0;

}

void print(int frame[], int nf)

{

for(int i=0; i<nf; i++)

{

if(frame[i] == -1)

printf("-- ");

else

printf("%2d ", frame[i]);

}

printf("|");

}

int findpos(int frame[],int m,int pages[],int cur,int n)

{

int i,j;

for(i=0;i<m;i++)

{

if(frame[i] == -1)

return i;

}

int pos[m]={0};

for(i=0;i<m;i++)

{

pos[i]=1e9;

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

{

if(pages[j]==frame[i])

{

pos[i]=j;

break;

}

}

}

int max=-1;

int Pos=-1;

for(i=0;i<m;i++)

{

if(pos[i]>max)

{

max=pos[i];

Pos=i;

}

}

return Pos;

}

void add(int frame[], int m, int n, int page[])

{

char a[100];

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

{

a[i]=to\_char(page[i]);

char p[10];

p[0] = a[i];

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,70,p);

printf(" %2d | ",page[i]);

if(!present(frame,m,page[i]))

{

int pos = findpos(frame,m,page,i,n);

frame[pos] = page[i];

setcolor(YELLOW);

circle(52+i\*80,132+pos\*43,19);

print(frame,m);

printf(" \*\n");

faults++;

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,130+43\*m,"\*");

draw(110,30+i\*80,frame,m);

continue;

}

print(frame,m);

printf("\n");

draw(110,30+i\*80,frame,m);

}

}

int main(){

int np,nf,i;

printf("\*\n\tOPT page replacement algorithm\n\*\n");

printf("\nEnter number of frames: ");

scanf("%d",&nf);

int frame[nf];

for(i=0;i<nf;i++)

{

frame[i]=-1;

}

int choice;

printf("\n\*\n\tEnter '1' :Create an array: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1\n");

printf(" \n\tEnter '2' :Custom\n");

printf(" \n\tEnter '3' :Input from file\n\*\n");

printf("\nchoose: ");

scanf("%d",&choice);

switch (choice)

{

case 1:

{

int np = 20;

int page[np]={7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};

printf("Number of pages: 12\n");

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

case 2:

{

printf("Enter number of pages: ");

scanf("%d",&np);

int page[np];

printf("\nEnter page value:\n");

for(i=0;i<np;i++)

{

printf("page[%d] = ",i);

scanf("%d",&page[i]);

}

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

case 3:

{

ifstream FileIn;

char name[30];

printf("\nEnter the path of file (Ex: 'D:\\Input.txt'): ");

scanf("%s", name);

FileIn.open(name,ios::in);

while(FileIn.fail()==true)

{

printf("\nFile not exist!\n");

printf("Please Enter correct the path of file: ");

scanf("%s", name);

FileIn.open(name,ios::in);

}

int page[100];

int np;

FileIn >> np;

for(i=0;i<np;i++)

{

FileIn >> page[i];

}

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page);

break;

}

}

printf("\nNumber of page faults : %d\n\n", faults);

while(!kbhit()) delay(1); // pause screen

return 0;

}

* **Kết quả:**

Chọn 2 và nhập chuỗi truy xuất, kết quả màn hình consoleText

Description automatically generated

Kết quả mô phỏng bằng thư viện graphics.h

Graphical user interface

Description automatically generated

**4.4 Thuật toán Second Chance**

* **Code:**

#include<graphics.h>

#include<bits/stdc++.h>

using namespace std;

int faults=0;

char to\_char(int n)

{

if(n>=0 && n<=9)

{

return n+48;

}

else

{

return ' ';

}

}

void pf(int x, int y, int n)

{

int f = faults;

for(int i=0;f>0;i++)

{

int tmp = f%10;

char a[10];

a[0] = to\_char(tmp);

settextstyle(8,0,3);

outtextxy(350-i\*15,130+43\*n+40,a);

f=f/10;

}

}

void draw(int x, int y, int frame[], int n)

{

settextstyle(8,0,4);

outtextxy(60,20,"Second Chance page replacement algorithm");

setcolor(GREEN);

settextstyle(8,0,3);

char a[100];

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

{

a[i]=to\_char(frame[i]);

}

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

{

char p[10];

p[0] = a[i];

rectangle(y,x+i\*43,y+43,x+(1+i)\*43);

if(i==0)

{

outtextxy(y+15,x+12,p);

}

else

{

outtextxy(y+15,x+12+i\*43,p);

}

}

settextstyle(8,0,3);

outtextxy(35,130+43\*n+40,"Number of page faults: ");

pf(x,y,n);

}

int present(int frame[], int nf, int page, int bit[])

{

for(int i=0; i<nf; i++)

if(page == frame[i])

{

bit[i] = 1;

return 1;

}

return 0;

}

void print(int frame[], int nf)

{

for(int i=0; i<nf; i++)

{

if(frame[i] == -1)

printf("-- ");

else

printf("%2d ", frame[i]);

}

printf("|");

}

int findpos(int frame[], int m, int x, int pos, int bit[])

{

while(true)

{

if(!bit[pos])

{

frame[pos] = x;

return (pos + 1) % m;

}

bit[pos] = 0;

pos = (pos + 1) % m;

}

}

void add(int frame[], int m, int n, int page[],int bit[])

{

int pos=0;

char a[100];

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

{

a[i]=to\_char(page[i]);

char p[10];

p[0] = a[i];

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,70,p);

printf(" %2d | ",page[i]);

if(!present(frame,m,page[i],bit))

{

setcolor(YELLOW);

circle(52+i\*80,132+pos\*43,19);

pos = findpos(frame,m,page[i],pos,bit);

print(frame,m);

printf(" \*\n");

faults ++;

setcolor(GREEN);

settextstyle(8,0,3);

outtextxy(45+i\*80,130+43\*m,"\*");

draw(110,30+i\*80,frame,m);

continue;

}

print(frame,m);

printf("\n");

draw(110,30+i\*80,frame,m);

}

}

int main()

{

int np,nf,i;

printf("\*\n\tSecond Chan page replacement algorithm\n\*\n");

printf("\nEnter number of frames: ");

scanf("%d",&nf);

int frame[nf], bit[nf];

for(i=0;i<nf;i++)

{

frame[i]=-1;

bit[i] = 0;

}

int choice;

printf("\n\*\n\tEnter '1' :Create an array: 0, 4, 1, 4, 2, 4, 3, 4, 2, 4, 0, 4, 1, 4, 2, 4, 3, 4\n");

printf(" \n\tEnter '2' :Custom\n");

printf(" \n\tEnter '3' :Input from file\n\*\n");

printf("\nchoose: ");

scanf("%d",&choice);

switch (choice)

{

case 1:

{

int np = 18;

int page[np]={0, 4, 1, 4, 2, 4, 3, 4, 2, 4, 0, 4, 1, 4, 2, 4, 3, 4};

printf("Number of pages: 12\n");

printf("\nposition of frame table after each request\n\n");

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

add(frame,nf,np,page,bit);

break;

}

case 2:

{

printf("Enter number of pages: ");

scanf("%d",&np);

int page[np];

printf("\nEnter page value:\n");

for(i=0;i<np;i++)

{

printf("page[%d] = ",i);

scanf("%d",&page[i]);

}

printf("\nposition of frame table after each request\n\n");

add(frame,nf,np,page,bit);

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

break;

}

case 3:

{

ifstream FileIn;

char name[30];

printf("\nEnter the path of file (Ex: 'D:\\Input.txt'): ");

scanf("%s", name);

FileIn.open(name,ios\_base::in);

if(FileIn.fail()==true)

{

printf("\nFile not exist!\n");

system("pause");

return 0;

}

int page[100];

int np;

FileIn >> np;

for(i=0;i<np;i++)

{

FileIn >> page[i];

}

printf("\nposition of frame table after each request\n\n");

add(frame,nf,np,page,bit);

initwindow(1400, 700);

setbkcolor(BLACK);

cleardevice();

break;

}

}

printf("\nNumber of page faults : %d\n\n", faults);

while(!kbhit()) delay(1); // pause screen

return 0;

}

* **Kết quả:**

Chọn 2 và nhập chuỗi truy xuất, kết quả màn hình Console

Text

Description automatically generated

Kết quả mô phỏng bằng thư viện graphics.h

Graphical user interface

Description automatically generated

**Chương IV: KẾT LUẬN**

1. **KẾT QUẢ ĐẠT ĐƯỢC**

Sau thời gian nghiên cứu và thực hiện đề tài, đến nay đề tài đã thực hiện được những yêu cầu như sau:

* Giao diện đồ họa, minh họa quá trình thay thế trang của từng giải thuật: FIFO, OPT, LRU và Second chance.
* Đánh dấu “ \* ” và khoanh tròn vào vị trí lỗi trang.
* Tính tổng số lượng lỗi trang (Page fault) của từng giải thuật.

1. **ƯU ĐIỂM**

* Nhập số khung trang và chuỗi truy xuất dễ dàng.
* Chương trình được viết bằng ngôn ngữ lập trình C nên việc lập trình khá cơ bản.
* Chương trình được xây dựng từ các hàm thuận lợi cho việc dùng lại và sửa lỗi.
* Chương trình xác định được những chỗ bị thay trang, đánh dấu “ \* ” dưới những khung trang có lỗi và in ra tổng số lỗi trang.

1. **HẠN CHẾ**

* Do lập trình bằng ngôn ngữ C và có sử dụng thêm thư viện đồ họa graphics.h nên để chạy chương trình cần phải cài đặt môi trường và cài đặt thêm thư viện đồ họa.
* Không có chức năng xuất dữ liệu minh họa ra file.
* Chưa minh họa được chi tiết quá trình thay thế (trang cũ đi ra, trang mới được thay vào).

1. **HƯỚNG PHÁT TRIỂN**

* Cài đặt chương mô phỏng trên nhiều ngôn ngữ lập trình khác.
* Thêm chức năng xuất dữ liệu mô phỏng ra file.
* Mô phỏng chi tiết quá trình thay trang.

1. **KẾT LUẬN**

Sau khi thực hiện đề tài đã cài đặt được chương trình có giao diện đồ họa, cho phép người dùng nhập vào khung trang, chuỗi truy xuất và minh họa quá trình thay thế trang của từng giải thuật: FIFO, OPT, LRU và Second Chance. Chương trình đánh dấu \* và khoanh tròn vào vị trí lỗi trang. Tính và in ra tổng số lỗi trang (Page fault) của từng giải thuật. Ta có thể so sánh tổng số lỗi trang của từng giải thuật và đưa ra nhận xét thuật toán nào tối ưu hơn.