Mục lục:

[Phần mở đầu 3](#_Toc513198547)

[Tổng quan đề tài 3](#_Toc513198548)

[Đề tài số: 17 3](#_Toc513198549)

[Mục đích: 3](#_Toc513198550)

[Phạm vi đề tài: 3](#_Toc513198551)

[Ý nghĩa báo cáo: 3](#_Toc513198552)

[Bố cục của báo cáo: 3](#_Toc513198553)

[Đóng góp tới đề tài của sinh viên: 3](#_Toc513198554)

[Sinh viên: Đinh Hoàng Nam MSSV: 2016.2793 3](#_Toc513198555)

[Sinh viên: Nguyễn Hữu Doanh MSSV: 2016.0631 4](#_Toc513198556)

[Phần nội dung 5](#_Toc513198557)

[Cơ sở lý thuyết: 5](#_Toc513198558)

[Cấu trúc và vấn đề truy nhập đĩa 5](#_Toc513198559)

[Các thuật toán điều phối 5](#_Toc513198560)

[Vấn đề lựa chọn thuật toán: 5](#_Toc513198561)

[Đặt vấn đề: 7](#_Toc513198562)

[Yêu cầu: 7](#_Toc513198563)

[Xác định bài toán: 7](#_Toc513198564)

[Một số cấu trúc dữ liệu sử dụng trong chương trình 8](#_Toc513198565)

[Một số cấu trúc dữ liệu đặc biệt (trong thư viện C++) 8](#_Toc513198566)

[Một số biến toàn cục được sử dụng trong chương trình 8](#_Toc513198567)

[Phân tích và thiết kế các thuật toán 9](#_Toc513198568)

[FCFS: 9](#_Toc513198569)

[SSTF 11](#_Toc513198570)

[SCAN: 14](#_Toc513198571)

[CSCAN: 16](#_Toc513198572)

[LOOK/CLOOK 18](#_Toc513198573)

[Một số hàm phụ trợ 19](#_Toc513198574)

[Hàm đọc dữ liệu 19](#_Toc513198575)

[Hàm khởi tạo 19](#_Toc513198576)

[Hàm in kết quả: 19](#_Toc513198577)

[Hàm chèn cylinders sau mỗi bước: 19](#_Toc513198578)

[Hàm in Menu 19](#_Toc513198579)

[Phần kết luận 20](#_Toc513198580)

[Kết luận chung 20](#_Toc513198581)

[Những kết quả đạt được: 20](#_Toc513198582)

[Những vấn đề còn tồn đọng: 20](#_Toc513198583)

[Hướng phát triển: 20](#_Toc513198584)

[Phụ lục: 21](#_Toc513198585)

[Mã nguồn của toàn bộ chương trình: 21](#_Toc513198586)

[Tài liệu tham khảo: 38](#_Toc513198587)

[Danh mục từ viết tắt: 38](#_Toc513198588)

# Phần mở đầu

## Tổng quan đề tài

### Đề tài số: 17

Tên đề tài: Xây dựng chương trình minh họa cho các giải thuật quản lý truy cập đĩa từ.

Mục đích:  
Trang bị thêm kiến thức về môn học.  
Rèn luyện kỹ năng tự học, kỹ năng lập trình, khả năng làm việc nhóm.

Phạm vi đề tài:  
Đề tài nghiên cứu trong phạm vi hẹp của môn học.  
Chương trình cài đặt 6 thuật toán cơ bản.

Ý nghĩa báo cáo:  
Báo cáo có thể dùng để làm tài liệu tham khảo trong quá trình học tập môn học.

Bố cục của báo cáo:  
Báo cáo được chia thành 3 phần, phần mở đầu gồm 2 chương, phần nội dung gồm 5 chương, phần kết luận gồm 3 chương.

Đóng góp tới đề tài của sinh viên:

Sinh viên: Đinh Hoàng Nam MSSV: 2016.2793  
Cài đặt giải thuật 1, 2, 3.  
Cài đặt tính năng insert cylinders sau mỗi bước.  
Tổng hợp và hoàn thiện mã nguồn.  
Viết báo cáo.

Sinh viên: Nguyễn Hữu Doanh MSSV: 2016.0631   
Cài đặt giải thuật 4, 5, 6.  
Kiểm thử chương trình.

# Phần nội dung

## Cơ sở lý thuyết:

Đĩa từ là một trong số các thiết bị vào ra của máy tính, cần được quản lý bởi Hệ điều hành.

Cấu trúc và vấn đề truy nhập đĩa  
Mô hình hóa như mảng một chiều các khối logic.  
Ánh xạ liên tiếp các khối logic tới các sector của đĩa.  
Hệ điều hành có trách nhiệm sử dụng hiệu quả phần cứng.  
Băng thông được tính dựa trên: tổng số bytes đã trao đổi và khoảng thời gian yêu cầu dịch vụ đầu tiên cho tới khi hoàn thành.  
Thời gian truy nhập gồm 2 phần: Thời gian định vị & Độ trễ quay.  
Mục đích: cực tiểu hóa thời gian định vị.  
Cần có Hàng đợi yêu cầu cho dịch vụ truy nhập đĩa.

Các thuật toán điều phối  
FCFS:First Come First Served  
SSTF: Shortest Seek Time First   
SCAN   
C-SCAN: Circular SCAN   
LOOK/C-LOOK

Vấn đề lựa chọn thuật toán:  
SSTF: Phổ biến, hiệu quả hơn FCFS.  
SCAN/C-SCAN hoạt động tốt hơn cho hệ thống có nhiều yêu cầu truy nhập đĩa vì không gặp vấn đề “starvation: chờ đợi quá lâu”.  
Hiệu quả của các thuật toán phụ thuộc mạnh về số lượng và kiểu các yêu cầu  
Yêu cầu truy xuất đĩa bị ảnh hưởng mạnh bởi các phương pháp phân phối đĩa cho file: Phân phối liên tục: đưa ra các y/cầu truy xuất lân cận nhau; Phân phối liên kết/ chỉ mục: có thể gồm các khối được phân bố rộng rãi trên đĩa.  
Thuật toán điều phối truy nhập đĩa có thể được viết như những modul riêng biệt của HĐH cho phép có thể thay thế bởi các thuật toán khác khi cần thiết.

Cả SSTF và LOOK đều có thể là lựa chọn hợp lý cho thuật toán mặc định.

## Đặt vấn đề:

Yêu cầu:   
Xây dựng chương trình minh họa cho các giải thuật quản lý truy cập đĩa từ.

### Xác định bài toán:

Input:  
Danh sách các cylinder cần truy cập (từ keyboard hoặc file)  
Giá trị lớn nhất của cylinder trên đĩa từ (max\_cld) (SCAN, CSCAN algorithm need input max\_cld)

Output:  
Danh sách thể hiện thứ tự truy cập các cylinder theo mỗi thuật toán (Order)  
Danh sách thể hiện chi phí dịch chuyển của mỗi bước (Cost)  
Tổng chi phí thực hiện theo mỗi thuật toán (Sum of cost)

Note:  
User có thể insert thêm cylinders sau mỗi bước thực hiện của thuật toán.  
Danh sách các cylinders nhập từ keyboard hoặc từ file cần kết thúc bởi -1  
Các thuật toán có thể output thêm một số thông tin phụ sau mỗi bước như Hàng đợi các yêu cầu truy nhập cylinders, hoặc Thứ tự sắp xếp các cylinders ở thời điểm hiện tại, giúp người dùng dễ theo dõi hơn.

## Một số cấu trúc dữ liệu sử dụng trong chương trình

### Một số cấu trúc dữ liệu đặc biệt (trong thư viện C++)

Véc-tơ (#include <vector>):  
mylistInput: dùng để chứa toàn bộ danh sách các cylinders input vào.  
order: chứa thứ tự truy cập các cylinders của thuật toán.  
dis: dùng để chứa khoảng cách giữa 2 cylinders ở từng bước.

Hàng đợi myQ (#include <queue>):  
Dùng để chứa các cylinder với tính chất FIFO  
Các phần tử còn lại trong hàng đợi là các cylinders đang chờ được truy cập  
Được sử dụng để cài đặt thuật toán FCFS

Tập hợp mys (#include <set>):  
Dùng để chứa danh sách các cylinders đang chờ được phục vụ ở từng thời điểm.  
Các cylinders đã phục vụ sẽ được loại bỏ khỏi tập hợp.  
Cấu trúc dữ liệu tập hợp cho phép tự động sắp xếp các giá trị theo thứ tự tăng dần.  
Cấu trúc dữ liệu tập hợp sẽ không cho phép chèn vào giá trị đang tồn tại trong tập hợp. (chỉ sau khi loại bỏ giá trị đó rồi thì mới có thể chèn lại vào được)  
Được sử dụng để cài đặt thuật toán: SSTF & SCAN & CSCAN & LOOK & CLOOK

Note: mylistInput, myQ, mys là các cấu trúc dữ liệu toàn cục, các thuật toán dùng đến các cấu trúc này sẽ được truyền vào bản sao tương ứng dsInput, Q, s để thao tác. Điều này đảm bảo cho việc thao tác trên thuật toán này không ảnh hưởng đến thuật toán khác.

### Một số biến toàn cục được sử dụng trong chương trình

max\_cld: giá trị lớn nhất của cylinder trong đĩa từ.  
cost: dùng để chứa tổng chi phí khi thực hiện thuật toán.

## Phân tích và thiết kế các thuật toán

FCFS:  
Input:  
Nhận bản sao Q từ hàng đợi myQ chứa danh sách các cylinders được input từ keyboard hoặc file theo thứ tự FIFO.   
Nhận bản sao dsInput từ mylistInput dùng để lưu trữ danh sách các cylinders được input và các cylinders được insert ở mỗi bước (nếu có).

Output: như trong phần đặt vấn đề.

Ý tưởng:  
Tại mỗi bước ta lấy một cylinder từ hàng đợi Q và:  
Bổ sung vào danh sách thứ tự các cylinders được truy nhập order  
Tính khoảng các giữa cylinder đó và cylinder kế trước  
Bổ sung khoảng cách vừa tính được vào danh sách dis  
Cập nhập giá trị của tổng chi phí += khoảng cách đó.  
Lặp lại quá trình trên đến khi hàng đợi Q rỗng.

#### Giả mã FSFS

Void FCFS( queue Q, vector dsInput )

{

int cur = Q.pop();

order.push(cld)

While ( Q khác rỗng )

{

int cld = Q.pop();

int d = | cld – cur |;

UpdateOuput( cld, d );

cur = cld;

InsertAfterEachStep(Q, dsInput);

}

PrintResult();  
}

### SSTF

Input:  
Nhận bản sao tập hợp s từ tập mys các cylinders được input từ keyboard hoặc file đã được sắp xếp theo thứ tự tăng dần.   
Nhận bản sao dsInput từ mylistInput dùng để lưu trữ danh sách các cylinders được input và các cylinders được insert ở mỗi bước (nếu có).

Output:  
như trong phần đặt vấn đề:

Ý tưởng:  
Sau khi đã sắp xếp các cylinders theo thứ tự tăng dần, tại mỗi bước ta thực hiện:  
So sánh khoảng cách từ cylinders hiện tại với 2 cylinders kề sát 2 bên, và lựa chọn cylinder nào có khoảng cách gần hơn. Đây chính là cylinders có khoảng cách ngắn nhất từ cylinder hiện tại.  
Bổ sung vào danh sách thứ tự các cylinders được truy nhập order  
Bổ sung khoảng cách vừa tính được vào danh sách dis  
Cập nhập giá trị của tổng chi phí += khoảng cách đó.  
Xóa cylinder đã được phục vụ ra khỏi tập hợp  
Lặp lại cho đến khi tập hợp chỉ còn 1 phần tử.

#### Giả mã SSTF:

Void STTF ( set s, vector dsInput )

{

int cur = cylinder xuất phát.

order.push( cur )

While( s.size() != 1)

{

if( cur ở cuối tập hợp)

{

int now = prev( cur );

d = cur - now;

UpdateOutput( now, d );

s.erase ( cur );

cur = now;

}

else if( cur ở đầu tập hợp )

{

int now = prev( cur );

d = now - cur;

UpdateOutput( now, d );

s.erase ( cur ); cur = now;

}

else if( cur ở giữa tập hợp)

{

prev = prev( cur );

post = next ( cur );

d\_prev = cur - prev;

d\_post = post - cur;

if ( d\_pre < d\_post )

UpdateOutput( prev, d\_prev );

else

UpdateOutput( post, post );

}

InsertAfterEachStep();

}

PrintResult();

}

### SCAN:

Input: Tập hợp s & danh sách dsInput.

Output: như trong phần đặt vấn đề.

Ý tưởng:  
B1:Sắp xếp các cylinders trong tập hợp theo thứ tự tăng dần  
B2:Bắt đầu từ cylinder xuất phát thực hiện vòng lặp for quét lần lượt đến các cylinder theo thứ tự giảm dần (cho tới khi tới giá trị 0)  
B3:Nếu vẫn còn cylinder trong tập hợp chưa được phục vụ, thực hiện vòng lặp for quét lần lượt đến các cylinder đó theo thứ tự tăng dần. (cho tới khi tới giá trị max\_cld)  
Lặp lại B2, B3 cho đến khi đã phục vụ hết các cylinder.

#### Giả mã SCAN:

void SCAN ( set s, vector dsInput )

{

cur = cylinder xuất phát

UpdateOutput( cur );

initial = prev( cur, 1);

while ( các cylinder vẫn chưa được phục vụ hết )

{

s.insert(0);

for( it = initial; it != s.begin(); it-- )

{

now = it;

d = cur - now;

UpdateOutput(now,d);

s.erase(cur); cur = now;

InsertAfterEachStep();

}

if( đã phục vụ xong )

break;

s.insert( max\_cld );

initial = next( cur );

for( it = initial; it ! s.end(); it++)

{

now = it;

d = now - cur;

UpdateOuput( now, d );

s.erase(cur); cur = now;

InsertAfterEachStep();

}

}

PrintResult();

}

### CSCAN:

Input: Tập hợp s và danh sách dsInput.

Output: Như phần đặt vấn đề.

Ý tưởng:

B1: Sắp xếp các cylinders theo thứ tự tăng dần.  
B2: Bắt đầu từ cylinders xuất phát, thực hiện vòng lặp for quét hết các cylinder theo thứ tự giảm dần. (cho tới khi tới giá trị 0)  
B3: Nếu vẫn còn cylinders chưa được phục vụ, gán giá trị khởi tạo vòng lặp ở B2 là cylinder lớn nhất của đĩa từ. (chính là giá trị của max\_cld)  
Lặp lại B2, B3 cho đến khi đã phục vụ hết các yêu cầu.

#### Giả mã CSCAN

void CSCAN ( set s, vector dsInput)

{

cur = cylinder xuất phát;

UpdateOutput( cur );

initial = prev(cur);

s.insert( 0 );

while ( chưa phục vụ xong )

{

for( it = initial; it != s.begin(); it-- )

{

now = it;

d = | cur - now |;

UpdateOutput(now, d);

s.erase( cur );

cur = it;

s.insert(0);

InsertAfterEachStep();

}

if ( đã phục vụ xong )

break;

else

{

s.insert( max\_cld );

initial = s.end();

}

}

PrintResult();

}

### LOOK/CLOOK

Input và Output tương tự như thuật toán SCAN và CSCAN

Ý tưởng:  
Tương tự như thuật toán SCAN và CSCAN, nhưng ta thay thế biên của các vòng lặp có giá trị 0 và giá trị max\_cld bởi cylinder nhỏ nhất và cylinder lớn nhất đang yêu cầu phục vụ.

## Một số hàm phụ trợ

Hàm đọc dữ liệu  
Void ReaData(void){}:  
thực hiện nhập dữ liệu từ bàn phím.

Void ReadData( string filename ) {}:  
thực hiện đọc dữ liệu từ file.

Hàm khởi tạo   
void init(){}  
thực hiện reset các vecto order, vector dis, giá trị cost về mặc định để xóa giá trị cũ của các thuật toán trước và khởi tạo các giá trị ban đầu trước khi thực hiện các thuật toán.

Hàm in kết quả:   
PrintResult() {}  
thực hiện printOutput.

Hàm chèn cylinders sau mỗi bước:  
void InsertToQueue(queue Q){} (dùng trong thuật toán FCFS)   
void InsertToSet(set s) {} (dùng trong các thuật toán còn lại)  
Cho phép user bổ sung thêm các cylinders sau mỗi bước, và có thể tắt chức năng này đi nếu không muốn xuất hiện nhiều lần.

Hàm in Menu  
void PrintMenu() {}:  
Print ra danh sách các thuật toán để user lựa chọn.

# Phần kết luận

## Kết luận chung

Những kết quả đạt được:  
Chương trình đã minh họa được một số giải thuật điều phối truy cập đĩa từ bằng màn hình console.  
Tính năng bổ sung các cylinders cần truy nhập sau mỗi bước giúp minh họa các giải thuật giống trong thực tế hơn.  
Chương trình sử dụng bảng chọn Menu giúp thân thiện với người dùng hơn.

Những vấn đề còn tồn đọng:  
Khi thực hiện nhiều thao tác insert cylinders sau mỗi bước, có thể kết quả chương trình sẽ khó theo dõi.  
Chương trình chưa tổ chức "error handling" tốt lắm, nếu user nhập không đúng định dạng có thể gây ra kết quả không mong muốn.

Hướng phát triển:  
Bổ sung thêm tính năng insert cylinders sau mỗi bước từ file  
Tối ưu mã nguồn, đảm bảo hoạt động tốt với kích thước input lớn  
Tổ chức xử lý "error handling" để đảm bảo chương trình hoạt động tốt hơn.

## Phụ lục:

### Mã nguồn của toàn bộ chương trình:

#include <stdio.h>

#include <conio.h>

#include <iostream>

#include <fstream>

#include <vector>

#include <queue>

#include <set>

#include <string>

#include <math.h>

using namespace std;

queue<int> myQ;

set<int> mys;

vector<int> mylistInput; // print Input

int max\_cld;

vector<int> order; // order

vector<int> dis; // distance

int cost; // sum cost

bool flag\_insert = false;

int count;

void ReadData()

{

cout << "\nInput cylinders want to access (Enter -1 to end): ";

int cld; cin >> cld;

while( cld != -1 )

{

myQ.push(cld);

mys.insert(cld);

mylistInput.push\_back(cld);

cin >> cld;

}

}

void ReadData(string filename)

{

cout << " Reading data from file... ";

ifstream f (filename);

int cld; f >> cld;

while( cld != -1 )

{

myQ.push(cld);

mys.insert(cld);

mylistInput.push\_back(cld);

f >> cld;

}

cout << "done!\n";

}

void init()

{

cost = 0;

dis.clear();

order.clear();

flag\_insert = false;

count = 0;

}

void PrintResult ( vector<int> dsInput )

{

cout << "\n\*\*\* Output \*\*\*\n";

cout << " Input: ";

for(int i = 0; i <= dsInput.size() -1 ; i++ )

cout << dsInput[i] << " ";

cout << "\n Order: ";

for( int i = 0; i <= order.size() - 2; i++)

printf("%3d --> ",order[i]);

cout << order[order.size() - 1];

cout << "\n Cost : ";

for( int i = 0; i <= dis.size() -1; i++)

printf(" %3d ",dis[i]);

cout << "\n Sum of cost: " << cost;

}

void PrintStepWithQueue( queue<int> Q )

{

cout << "\n--Step " << ++count << "--" << endl;

cout << " Order:";

for( int i = 0; i <= order.size() - 1; i++)

{

cout <<" " << order[i] << " --> ";

}

cout << "\n Queue:";

while( !Q.empty() )

{

cout << " " << Q.front();

Q.pop();

}

}

void PrintStepWithSet( set<int> s )

{

cout << "\n--Step " << ++count << "--"<< endl;

cout << " Order:";

for( int i = 0; i <= order.size() - 1; i++)

{

cout <<" " << order[i] << " --> ";

}

cout << "\n Set:";

for(set<int>::iterator it = s.begin(); it != s.end(); it++)

cout << " " << \*it;

}

void InsertToQueue( queue<int>& Q, vector<int>& dsInput )

{

if( flag\_insert == false)

{

PrintStepWithQueue( Q );

cout << "\nDo you want to insert cylinders? (y/n) (c to cancle, no more insert) ";

char ch; cin >> ch;

if( ch == 'y' )

{

cout << "====> Input cylinders want to access (Enter -1 to end): ";

int cld; cin >> cld;

while( cld != -1 )

{

Q.push(cld);

dsInput.push\_back(cld);

cin >> cld;

}

}

else if( ch == 'c')

flag\_insert = true;

}

}

void InsertToSet( set<int>& s, vector<int>& dsInput )

{

if( flag\_insert == false)

{

PrintStepWithSet( s );

cout << "\nDo you want to insert cylinders? (y/n) (c to cancle, no more insert) ";

char ch; cin >> ch;

if( ch == 'y' )

{

cout << "====> Input cylinders want to access (Enter -1 to end): ";

int cld; cin >> cld;

while( cld != -1 )

{

s.insert(cld);

dsInput.push\_back(cld);

cin >> cld;

}

}

else if( ch == 'c')

flag\_insert = true;

}

}

void FCFS( queue<int> Q, vector<int> dsInput )

{

init();

cout << "\n\nFCFS:\n";

int cur = Q.front(); Q.pop();

order.push\_back(cur);

while( !Q.empty() )

{

int cld = Q.front(); Q.pop();

order.push\_back( cld );

int d = abs( cld - cur );

dis.push\_back( d );

cur = cld;

cost += d;

InsertToQueue(Q, dsInput);

}

PrintResult( dsInput );

}

void SSTF( set<int> s, vector<int> dsInput )

{

init();

cout << "\n\nSSTF:\n";

set<int>::iterator cur = s.find( dsInput[0] ); order.push\_back(\*cur);

int d, d\_pre, d\_post;

while( s.size() != 1 )

{

if( cur == s.begin() && cur != --s.end() )

{

set<int>::iterator post = next(cur,1);

order.push\_back( \*post );

d = \*post - \*cur;

dis.push\_back( d );

cost += d;

s.erase( cur );

cur = post;

}

else if( cur == --s.end() && cur != s.begin() )

{

set<int>::iterator pre = prev(cur,1);

order.push\_back( \* pre );

d = \* cur - \*pre;

dis.push\_back( d );

cost += d;

s.erase( cur );

cur = pre;

}

else

{

set<int>::iterator pre = prev(cur,1);

set<int>::iterator post = next(cur,1);

d\_pre = \*cur - \*pre;

d\_post = \*post - \*cur;

if( d\_pre < d\_post )

{

order.push\_back( \*pre );

dis.push\_back( d\_pre );

cost += d\_pre;

s.erase ( cur );

cur = pre;

}

else

{

order.push\_back( \*post );

dis.push\_back( d\_post );

cost+= d\_post;

s.erase( cur );

cur = post;

}

}

InsertToSet( s, dsInput);

}

PrintResult( dsInput );

}

void SCAN ( set<int> s, vector<int> dsInput )

{

cout << "\n\nSCAN:\n";

init();

set<int>::iterator cur = s.find( dsInput[0] );

order.push\_back(\*cur);

bool flag = false;

while(1)

{

s.insert(0);

set<int>::iterator initial = prev(cur,1);

for(set<int>::iterator it = initial; it != --s.begin(); it--)

{

if( s.size() == 2)

{

flag = true;

break;

}

int now = \*it;

int d = \*cur - now;

order.push\_back(now);

dis.push\_back(d);

cost+=d;

s.erase(cur);

cur = it;

InsertToSet( s, dsInput);

}

if( flag == true)

break;

s.insert( max\_cld );

initial = next( cur, 1);

for(set<int>::iterator it = initial; it != s.end(); it++)

{

if( s.size() == 2)

{

flag = true;

break;

}

int now = \*it;

int d = now - \*cur;

order.push\_back(now);

dis.push\_back(d);

cost+=d;

s.erase(cur);

cur = it;

InsertToSet( s, dsInput);

}

if( flag == true)

break;

}

PrintResult( dsInput );

}

void CSCAN(set<int> s, vector<int> dsInput)

{

cout <<"\n\nCSCAN:\n";

init();

set<int>::iterator cur = s.find( dsInput[0] );

order.push\_back( \*cur );

set<int>::iterator initial = prev(cur,1);

bool flag = false;

s.insert(0);

while(1)

{

for(set<int>::iterator it = initial; it != --s.begin(); it--)

{

if( s.size() == 2 && \*(prev( cur, 1)) == 0 )

{

flag = true;

break;

}

int now = \*it;

int d = abs(\*cur - now);

order.push\_back(now);

dis.push\_back(d);

cost+=d;

s.erase( cur );

cur = it;

s.insert(0);

InsertToSet( s, dsInput);

}

if( flag == true)

break;

s.insert( max\_cld );

initial = --s.end();

}

PrintResult( dsInput );

}

void LOOK( set<int> s, vector<int> dsInput )

{

cout << "\n\nLOOK:\n";

init();

set<int>::iterator cur = s.find( dsInput[0] );

order.push\_back(\*cur);

while ( s.size() != 1 )

{

set<int>::iterator initial = prev(cur, 1);

for(set<int>::iterator it = initial; it != --s.begin(); it--)

{

int now = \*it;

int d = \*cur - now;

order.push\_back(now);

dis.push\_back(d);

cost+=d;

s.erase(cur);

cur = it;

InsertToSet( s, dsInput);

}

if( s.size() == 1 )

break;

initial = next( cur, 1 );

for( set<int>::iterator it = initial; it != s.end(); it++)

{

int now = \*it;

int d = now - \*cur;

order.push\_back(now);

dis.push\_back(d);

cost+=d;

s.erase(cur);

cur = it;

InsertToSet( s, dsInput);

}

}

PrintResult( dsInput );

}

void CLOOK( set<int> s, vector<int> dsInput )

{

cout << "\n\nCLOOK:\n";

init();

set<int>::iterator cur = s.find( dsInput[0] );

order.push\_back( \*cur );

set<int>::iterator initial = prev( cur, 1 );

while( s.size() != 1)

{

for(set<int>::iterator it = initial; it != --s.begin(); it --)

{

int now = \*it;

int d = abs(now - \*cur);

order.push\_back(now);

dis.push\_back(d);

cost+=d;

s.erase(cur);

cur = it;

InsertToSet( s, dsInput);

}

if( s.size() == 1)

break;

initial = prev(s.end(),1);

}

PrintResult( dsInput );

}

void Greeting()

{

printf("De tai 17: Chuong trinh minh hoa cho cac giai thuat quan ly truy cap dia tu.");

printf("\n\nInput:");

printf("\n -danh sach cac cylinders can truy cap (tu keyboard hoac file)");

printf("\n -gia tri max cua cylinder (max\_cld) (SCAN, CSCAN algorithm need input max\_cld)");

printf("\nOutput:");

printf("\n -thu tu cac cylinders duoc truy nhap (Order)");

printf("\n -chi phi cua moi step (Cost)");

printf("\n -tong chi phi theo thuat toan (Sum of cost)");

printf("\n\nNote:");

printf("\n -danh sach cylinders nhap tu keyboard hoac file can ket thuc boi -1");

printf("\n -user co the insert them cylinders sau moi step");

printf("\n -cac thuat toan co the output them mot so thong tin phu");

}

void SelectInput()

{

printf("\n\nInput danh sach cylinders:\n 1. keyboard\n 2. file\n");

printf(" your selection: ");

int x; cin >> x;

switch( x )

{

case 1:

{

ReadData();

} break;

case 2:

{

cout << " input file name: ";

string s; cin >> s;

ReadData( s );

} break;

}

printf("Input max\_cld = ");

cin >> max\_cld;

}

void PrintMenu()

{

int x;

char ch;

do

{

printf("\n=== Menu ===");

printf("\n1. FCFS\n2. SSTF\n3. SCAN\n4. CSCAN\n5. LOOK\n6. CLOOK\n7. all");

printf("\n your selection: ");

cin >> x;

switch ( x )

{

case 1: FCFS( myQ, mylistInput ); break;

case 2: SSTF( mys, mylistInput ); break;

case 3: SCAN( mys, mylistInput ); break;

case 4: CSCAN( mys, mylistInput ); break;

case 5: LOOK( mys, mylistInput ); break;

case 6: CLOOK( mys, mylistInput ); break;

case 7: { FCFS( myQ, mylistInput ); SSTF( mys, mylistInput ); SCAN( mys, mylistInput ); CSCAN( mys, mylistInput ); LOOK( mys, mylistInput ); CLOOK( mys, mylistInput ); } break;

}

printf("\n\nDo you want to start menu again? (y/n) ");

cin >> ch;

}while( ch == 'y');

cout << "\nThank you very much!";

cout << "\nSV thuc hien:\n Dinh Hoang Nam MSSV: 2016.2793\n Nguyen Huu Doanh MSSV: 2016.0631";

printf("\nPress any keys to exit..."); getch();

}

int main()

{

Greeting();

SelectInput();

PrintMenu();

return 0;

## Tài liệu tham khảo:

Slide bài giảng môn học “Nguyên lý các hệ điều hành” – Thầy Đỗ Quốc Huy  
Thư viện C++: <http://www.cplusplus.com/reference>

## Danh mục từ viết tắt:

FCFS:First Come First Served  
SSTF: Shortest Seek Time First   
C-SCAN: Circular SCAN