**PHƯƠNG PHÁP LUẬN LẬP TRÌNH**

1. **KHÁI NIỆM VỀ NNLT:**

Tất cả các NNLT đều bắt đầu từ ngôn ngữ Bare Bone. Mọi NNLT đều có khuynh hướng tiến dần đến ngôn ngữ tự nhiên và ngày càng đa dạng hoá (về ngôn ngữ và phong cách)

Hiện tại lập trình có 4 hướng:

1. Lập trình thủ tục : ta cải tiến để tránh sự nhập nhằng; chương trình phải trong suốt như toán học.

f(x) + g(x) = g(x) + f(x) (toán)

f(x) + g(x) <> g(x) + f(x) (thủ tục)

Vì hiệu ứng lề

1. Lập trình hướng đối tượng: mang tính chất 🡪 phân cấp và kế thừa

🡪 độc lập

1. Lập trình logic: mô tả bài toán và các luật, sau đó để cho chương trình dịch và xử lý.
2. Lập trình genetic: lập trình theo kiểu tự nhiên; Từ 1 bài toán bất kỳ, ta cài đặt nó như bài toán sinh học; chương trình dần tối ưu cho tới khi cảm thấy là đạt yêu cầu thì dừng.

Các NNLT sẽ tiếp tục phát triển song song, chứ không có loại NNLT nào tồn tại độc lập, và tối ưu hơn.

**II.** **PHƯƠNG PHÁP LUẬN LẬP TRÌNH**

Một chương trình được gọi là tốt khi nó thỏa các điều kiện sau:

* Tính dừng
* Tính đúng đắn
* Tính an toàn, ổn định
* Tính tối ưu

1. **Tính dừng**: chương trình phải đảm bảo dừng tại 1 thời điểm nào đó.

Xét chương trình sau:

void main() {

int x;

scanf(“%d”, &x);

while (x>1)

{

if (( x/ 2) \* 2 == x) x= x/2;

else x=3\*x+1;

printf(“%8d”,x);

}

}

Bạn hãy kiểm tra xem chương trình trên có dừng hay không ?

**2 Tính đúng đắn:** Trong thực tiễn, người ta đã ước tính có đến khoảng 27000 tỷ lệnh, 1.7 tỷ hàm (tính đến 1995). Khi vào ứng dụng, vẫn còn 5🡪7% lỗi chưa tìm ra; các lỗi này sau 5 năm khi đưa vào sử dụng thì có 85🡪90% của 5% lỗi tiềm ẩn được phát hiện. Có 3 dạng lỗi :

* Dạng 1: Không ảnh hưởng tới kết quả; chỉ làm ta khó chịu về hình thức
* Dạng 2: Lỗi về nội dung
* Dạng 3: Lỗi về hình thức lẫn nội dung

Lưu ý: Chi phí để bảo trì cho 1 dòng lệnh sẽ tốn kém gấp 2 cho đến 3 lần chi phí cho việc viết dòng lệnh đó.

Các nguyên lý chung:

1. ***Nguyên lý sai sót của chương trình***

**Bài toán 1**: Tìm chỗ sai của chương trình sau:

Input : số nguyên x, số nguyên y

r = x; q=0;

while r>y do

{

r=r -y;

q=q+1;

}

Kết quả : q : thương số, r:dư số.

Kiểm tra: - Điều kiện của phép chia y<>0:

Nếu y = 0 vòng lặp sẽ không sai.

- Các cận : với x kết quả sai vì dư r = x và thương q = 0.

**Bài toán 2**: Bài toán nhân 2 ma trận

Procedure gt2(a,b:day;var c:day; n:integer);

var i,ii,jj,j,k,kk:integer;

begin

i:=1;

while i<n do

begin

ii:=i+1;

j:=1;

while j<n do

begin

jj:=j+1;

c[i,j]:=0;

c[i,jj]:=0;

c[ii,j]:=0;

c[ii,jj]:=0;

k:=1;

while k<n do

begin kk:=k+1;

c[i,j]:=c[i,j] + a[i,k]\*b[k,j] + a[i,kk]\*b[kk,j];

c[i,jj]:=c[i,jj] + a[i,k]\*b[k,jj]+ a[i,kk]\*b[kk,jj];

c[ii,j]:=c[ii,j] + a[ii,k]\*b[k,j]+ a[ii,kk]\*b[kk,j];

c[ii,jj]:=c[ii,jj] + a[ii,k]\*b[k,jj]+a[ii,kk]\*b[kk,jj];

k:=k+2;

end;

j:=j+2;

end;

i:=i+2;

end;

end;

Nguyên tắc của thuật toán này thay vì tính một ô kết quả sau khi ra khỏi vòng lặp trong thì nó tính cùng lúc 4 ô kết quả.

Kiểm tra theo kích thước của ma trận vuông:

Nếu n chẳn thì kết quả đúng nhưng với n lẻ thì kết quả sai do bước tăng của các biến đếm là 2.

***2. Nguyên lý kiểm chứng logic chương trình theo mô hình nghịch đảo:***

Thuật toán nhân Ấn Độ:

Input : x,y

Output : z= x.y

Thuật toán:

Nhập x,y

if x lẻ then z:=y else z:=0

while x>1 do

begin

x:=x div 2;

y:=y \*2;

if x lẻ then z:=z+y;

end;

Chương trình tính thuật toán nhân Ấn Độ và kiểm chứng theo mô hình nghịch đảo:

Program nhan\_ando;

var

x,y,k,m,n:word;

begin

Repeat write('Nhap gia tri x :');readln(x); until x>=0;

Repeat write('Nhap gia tri y :');readln(y); until y>=0;

m:=x;

n:=y;

if (x mod 2)=1 then k:=y else k:=0;

if y<>0 then

begin

while x>1 do

begin

x :=x shr 1;

y :=y shl 1;

if (x mod 2)=1) then k:=k+y;

end;

writeln('x \* y = ',k);

(\* doan chuong trinh nghich dao\*)

if ((k/n)=m) and ((k/m)=n) then writeln('Thuat toan dung')

else writeln('Thuat toan sai!');

readln;

end;

end.

**II. TỐI ƯU HÓA CHƯƠNG TRÌNH**

***1. Tinh chế cách viết*:**

**Bài toán 1** : ***Thuật toán sắp xếp mảng Chọn trực tiếp***

|  |  |
| --- | --- |
| ***Thuật toán 1*** | ***Thuật toán 2*** |
|  | Tối ưu chương trình bằng cách tăng số lệnh thực hiện trong một bước vòng lặp để giảm số bước vòng lặp. |
| i:=1;  while i<=n do  j:=i+1;  while j<=n do  if a[i]>a[j] then  doicho(a[i],a[j]);  j:=j+1;  endwhile;  i:=i+1;  endwhile; | i:=1;  while i<=n do  j:=i+1;  while j<=n do  if a[i]>a[j] then doicho(a[i],a[j]);  if (j+1)<=n and a[i]>a[j+1] then  doicho(a[i],a[j+1]);  j:=j+2;  endwhile;  i:=i+1;  endwhile; |

*Sau đây là chương trình minh họa giải thuật sắp xếp mảng sau khi đã cải tiến thuật toán cũ, trong chương trình có sử dụng thêm 2 biến phụ là :*

*gan: để đếm tổng số phép gán của thuật toán*

*ss : để đếm số phép so sánh của thuật toán*

**program random\_gen;**

uses crt;

type dopt = record

ss,gan:integer;

end;

var x:array[1..100] of integer;

pthu:array[1..50] of dopt;

t,i,j,gan,ss,n,k1: integer;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

procedure swapint(var a,b:integer); /\* Đổi chổ a và b\*/

var t:integer;

begin

t:=a;

a:=b;

b:=t;

end;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function randint(a,b:integer):integer;

begin

randint :=a + random(b-a+1);

end;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

begin

clrscr;

write('Nhap so phan tu cua day :');readln(n);

write('So lan thu :');readln(k1);

randomize;

for t:=1 to k1 do

begin

for i:=1 to n do x[i]:=i;

for i:=1 to n do swapint(x[i],x[randint(i,n)]);

writeln;

Writeln('\* Lan thu ',t,' \*');

for i:=1 to n do write('x[',i,']=',x[i],' ');

gan:=0;

ss:=0;

i:=1;

gan:=gan+1;

ss:=ss+1;

while i<=n-1 do

begin j:=i+1;

gan:=gan+1;

ss:=ss+1;

while j<=n do

begin

ss:=ss+1;

if x[i]>x[j] then

begin

swapint(x[i],x[j]);

gan:=gan+3;

end;

if (j+1)<=n and x[i]>x[j+1] then

begin

swapint(x[i],x[j+1]);

gan:=gan+3;

end;

j:=j+2;

gan:=gan+1;

ss:=ss+1;

end;

i:=i+1;

gan:=gan+1;

ss:=ss+1;

end;

pthu[t].ss:=ss;

pthu[t].gan:=gan;

writeln;

writeln('So phep ss = ',pthu[t].ss,' So phep gan = ',pthu[t].gan);

writeln;

writeln('Nhan phim Enter de tiep tuc!');

readln;

end;

end.

**Bài toán 2** : Tính biểu thức E = 1/1! + 3/2! + 5/3! + … + (2n-1)/n!

|  |  |
| --- | --- |
| Giải thuật A | Giải thuật B |
|  | Tối ưu chương trình bằng cách giảm các vòng lặp lồng nhau. |
| E:=0;  For i:=1 to n do  Begin  Gt:=1;  For j:=1 to i do Gt:=Gt\*j;  E:= E +(2\*i-1)/Gt;  End; | E:=0;  Gt:=1;  For i:=1 to n do  Begin  Gt:=Gt\*i;  E:= E +(2\*i-1)/Gt;  End; |

Độ phức tạp của giải thuật A là n+(1+n)n/2 +1 có cấp O(n2)

Độ phức tạp của giải thuật B là 2\*n +2 có cấp là O(n)

Ở giải thuật B, ta áp dụng nguyên lý làm nhiều công việc có thể có trong một vòng lặp.

**Bài toán 3**: Nhân 2 ma trận vuông

|  |  |
| --- | --- |
| **Thuật toán A** | **Thuật toán B** |
|  | Tối ưu chương trình bằng cách tăngsố lệnh thực hiện trong một bước vòng lặp để giảm số bước vòng lặp. |
| procedure gt1(a,b:day;var c:day);  var i,j,k:integer;  begin  for i:=1 to n do  for j:=1 to n do  begin c[i,j]:=0;  for k:=1 to n do  c[i,j]:= c[i,j] + a[i,k]\*b[k,j];  end;  end; | procedure gt2(a,b:day;var c:day;n:integer);  var i,ii,jj,j,k,kk:integer;  begin  i:=1;  if n mod 2 = 1 then inc(n);  while i<n do  begin  ii:=i+1;  j:=1;  while j<n do  begin  jj:=j+1;  c[i,j]:=0;  c[i,jj]:=0;  c[ii,j]:=0;  c[ii,jj]:=0;  k:=1;  while k<n do  begin  kk:=k+1;  c[i,j]:=c[i,j] + a[i,k]\*b[k,j] + a[i,kk]\*b[kk,j];  c[i,jj]:=c[i,jj] + a[i,k]\*b[k,jj] + a[i,kk]\*b[kk,jj];  c[ii,j]:=c[ii,j] + a[ii,k]\*b[k,j] + a[ii,kk]\*b[kk,j];  c[ii,jj]:=c[ii,jj] + a[ii,k]\*b[k,jj] + a[ii,kk]\*b[kk,jj];  k:=k+2;  end;  j:=j+2;  end;  i:=i+2;  end;  end; |

Văn bản chương trình nhân ma trận tối ưu :

program nhanmatran;

type day = array[1..10,1..10] of real;

var a,b,c:day;

n:integer;

bool:boolean;

procedure khoidong;

var i,j:integer;

begin

for i:=1 to 10 do

for j:=1 to 10 do

begin

a[i,j]:=0;

b[i,j]:=0;

c[i,j]:=0;

end;

end;

procedure nhap(var m:day);

var i,j:integer;

begin

for i:=1 to n do

for j:=1 to n do

begin

writeln('M[',i,',',j,']=');

readln(m[i,j]);

end;

end;

procedure gt1(a,b:day;var c:day);

var i,j,k:integer;

begin

for i:=1 to n do

for j:=1 to n do

begin c[i,j]:=0;

for k:=1 to n do

c[i,j]:= c[i,j] + a[i,k]\*b[k,j];

end;

end;

procedure gt2(a,b:day;var c:day;n:integer);

var i,ii,jj,j,k,kk:integer;

begin

i:=1;

if n mod 2 = 1 then inc(n);

while i<n do

begin

ii:=i+1;

j:=1;

while j<n do

begin

jj:=j+1;

c[i,j]:=0;

c[i,jj]:=0;

c[ii,j]:=0;

c[ii,jj]:=0;

k:=1;

while k<n do

begin kk:=k+1;

c[i,j]:=c[i,j] + a[i,k]\*b[k,j] + a[i,kk]\*b[kk,j];

c[i,jj]:=c[i,jj] + a[i,k]\*b[k,jj] + a[i,kk]\*b[kk,jj];

c[ii,j]:=c[ii,j] + a[ii,k]\*b[k,j] + a[ii,kk]\*b[kk,j];

c[ii,jj]:=c[ii,jj] + a[ii,k]\*b[k,jj] + a[ii,kk]\*b[kk,jj];

k:=k+2;

end;

j:=j+2;

end;

i:=i+2;

end;

end;

procedure xuat(m:day);

var i,j:integer;

begin

for i:=1 to n do

begin

for j:=1 to n do

write(m[i,j]:8:2);

writeln;

end;

end;

BEGIN

writeln('Nhap kich thuoc ma tran vuong :');

readln(n);

nhap(a);

nhap(b);

gt1(a,b,c);

xuat(c);

gt2(a,b,c,n);

writeln;

xuat(c);

readln;

END.

Chương trình này đã khắc phục được phần sai ở phần 1 bằng cách:

-Khởi động ma trận vuông, kích thước của ma trận sử dụng phải lớn hơn ma trận sử dụng một dòng, một cột.

- Kiểm tra nếu n là lẽ thì tăng n lên 1.

**III. TINH CHẾ THUẬT TOÁN**

**Bài toán 1** : Chương trình tính giá trị của đa thức:

Y = an.xn + an-1. xn-1 + … + a1.x + a0=

= (an(((x+an-1) + ……. )x+ a0

|  |  |
| --- | --- |
| **Giải thuật A** | **Giải thuật B** |
| Y:= a[0];  Xtam:=1;  For i:=1 to n do  Begin Xtam:=X \* Xtam;  Ytam:= a[i ]\*Xtam;  Y:= y + Ytam;  End; | Y:= a[n];  For i:=n-1 downto 0 do  Begin  Ytam:=X\*Y;  Y:= Ytam + a[i];  End; |

Giải thuật A có độ phức tạp : 3\*n +2;

Phân tích : Y= ((((an.X+ an-1).X + an-2)X…)+ á1).X + a0 ta được thuật toán B

Giải thuật B có độ phức tạp là : 2n+1 tức nhanh hơn giải thuật A

Giải thuật B tinh giảm được số phép gán trong vòng lặp. Ngoài ra cả hai giải thuật đều áp dụng phương pháp gán đơn giản.

**Bài toán 2**: Phân loại ký tự

Cho một chuỗi gồm một triệu ký tự, hãy phân loại mỗi ký tự theo 4 kiểu sau: Kiểu chữ thường, kiểu chữ hoa, chữ số và kiểu khác.

Một cách giải thông thường là xác định một ký tự nào đó trong bảng mã ASCII thuộc kiểu "khac" bằng cách sử dụng 6 lần so sánh như sau:

If (kytu>='a') and (kytu<='z') then ký tự là ký tự thường

Else If (kytu>='A') and (kytu<='Z') then ký tự là ký tự hoa

Else If (kytu>='0') and (kytu<='9') then ký tự là ký tự số

Else Ký tự là ký tự khác.

Để tối ưu hóa chương trình, ta thiết lập một bảng truy cập như sau:

for c:=chr(0) to chr(255) do

if c IN ['a'..'z'] then TypeTable[c]:=Kieu\_chu\_thuong

else if c IN ['A'..'Z'] then TypeTable[c]:=Kieu\_chu\_hoa

else if c IN ['0'..'9'] then TypeTable[c]:=Kieu\_chu\_so

else TypeTable[c]:=Kieu\_khac

Như vậy kiểu của ký tự c là TypeTable[c] tránh được việc so sánh.

*Văn bản chương trình:*

Program Phan\_loai\_kieu\_ky\_tu;

Uses crt,dos;

Const Kieu\_khac='Khac';

Kieu\_chu\_thuong='chu thuong';

Kieu\_chu\_hoa='chu hoa';

Kieu\_chu\_so='chu so';

Var TypeTable : array[char] of string[10];

Filename: PathStr;

**Procedure Tao\_bang;**

Var c: char;

begin

for c:=chr(0) to chr(255) do

if c IN ['a'..'z'] then TypeTable[c]:=Kieu\_chu\_thuong

else if c IN ['A'..'Z'] then TypeTable[c]:=Kieu\_chu\_hoa

else if c IN ['0'..'9'] then TypeTable[c]:=Kieu\_chu\_so

else TypeTable[c]:=Kieu\_khac

end;

**Procedure Phan\_loai\_ky\_tu(Filename:PathStr);**

Var S: String; f: text; i: byte;

Begin

assign(f,filename);

{$I-}

reset(f);

{$I+}

If IoResult<> 0 then begin

writeln(#13#10'File ',filename,' khong co');

readln;

halt(0);

end;

While not eof(f) do

begin

readln(f,s);

For i:=1 to length(S) do

writeln(S[i], ' : ', TypeTable[s[i]]);

end;

close(f);

End;

Begin

clrscr;

Tao\_bang;

write('Ten file de thu phan loai : '); readln(filename);

If filename <> '' then

Phan\_loai\_ky\_tu(Filename);

readln;

End.

**Bài toán 3**: Tính số bit

Cho một file gồm một triệu từ 32 bit, hãy đếm tong số bit 1 trong file.

Một lời giải thông thường là sử dụng vòng lặp thực hiện 32 phép toán shift và phép toán logic and.

Để tối ưu hóa, ta thiết lập một bảng truy cập Counttable[0..255] chứa số bit 1 như sau:

Counttable[0]:=0;

Counttable[1]:=1;

Counttable[2]:=1;

…

Counttable[255]:=8;

Như vậy để tính số bit 1 trong từ W (4 bytes), ta dùng lệnh:

WordCount:= CountTable[W and mask]

+ CountTable[(W Rshift 8) and mask]

+ CountTable[(W Rshift 16) and mask]

+ CountTable[(W Rshift 24) and mask]

*Văn bản chương trình:*

Program So\_bit\_1\_cua\_mot\_trieu\_tu;

Uses Crt;

const mask:byte=$FF;

sotu=16000;

Type Bang\_dem = array[0..255] of byte;

Danh\_sach\_tu=array[1..sotu] of LongInt ;

Type str32= String[32];

Var i,sobit1, n:LongInt;

Count: Bang\_dem;

DStu: Danh\_sach\_tu;

**Function Dec\_Bin(n:LongInt):Str32;**

Var kt:String[1]; S: Str32;

Begin

S:='';

repeat

Str(n mod 2,kt);

S:=kt+S;

n:=n div 2;

until n =0;

Dec\_Bin:=S;

end;

**Procedure Xay\_dung\_Bang\_Count(Var Count:Bang\_dem);**

Var i,j:byte; S:Str32;

Begin

For i:=0 to 255 do

Begin

Count[i]:=0;

S:=Dec\_Bin(i);

For j:=1 to Length(S) do

If S[j]='1' then Inc (Count[i]);

End;

End;

**Procedure Phat\_sinh\_tu(Var DStu:Danh\_sach\_tu);**

Var i:Longint;

Begin

Randomize;

i:=1;

Repeat

DStu[i]:=Random($FFFF)\*2;

if dstu[i] < 0 then halt(0);

inc(i);

until i>sotu;

End;

**Procedure In\_ds\_tu(Var dstu:danh\_sach\_tu);**

Var i: LongInt;

Begin

for i:=1 to sotu do

writeln(Dec\_Bin(DStu[i]));

End;

Begin

clrscr; sobit1:=0;

Phat\_sinh\_tu(DStu);

Xay\_dung\_Bang\_Count(Count);

(\* In\_ds\_tu(dstu); \*)

For i:=1 to sotu do

begin

n:=Dstu[i];

sobit1:=sobit1+ Count[n and mask] + Count[(n shr 8) and mask] +

Count[(n shr 16) and mask] +Count[(n shr 24) and mask];

end;

writeln(#13#10,'So bit 1 trong ',sotu,' tu = ',sobit1);

readln;

end.

**Bài toán 4**:

Sắp xếp một dãy số nguyên 20000 số phân biệt khác nhau (các số nguyên đó có trị từ 1..30000) mà bộ nhớ trong chỉ có 1200 từ (4 bytes/từ).

Với bộ nhớ trong hạn chế theo như đề bài, ta không thể lưu trữ hết 20000 số nguyên vào bộ nhớ trong để áp dụng các giải thuật sort nội. Trong trường hợp này, một giải pháp có thể được là lưu 20000 số vào bộ nhớ ngoài và áp dụng giải thuật sort ngoại để sắp xếp chúng. Với cách thức lưu trữ và sắp xếp này, ta sẽ tốn thời gian để truy xuất dữ liệu và sắp xếp.

Do tính chất của bài toán là 20000 số nguyên từ 1..30000 và phân biệt khác nhau, nên một giải thuật tối ưu về cả thời gian và bộ nhớ được đưa ra ở đây là:

- Ta dùng một mảng A có 3750 bytes ≅ 938 từ (4 bytes) chứa 30000 số từ 1..30000.

- Cho vòng lặp chạy từ 1 đến 20000

+ Nhập vào số n ∈ [1..30000]

+ Ta bật bit tương ứng với số n trong A lên 1

- Sau khi nhập xong 20000 số , dãy số của ta cũng đã được sắp xong.

*Văn bản chương trình:*

Program Sap\_xep\_20000\_so\_trong\_3750\_byte; (\* moi so chi xuat hien 1 lan \*)

uses crt;

Const Sobyte=3750; (\* = 30000/8\*)

So\_Max = 30000;

Var A: array [1..Sobyte] of byte; (\* chứa 20000 so tu 1 den So\_Max\*)

X: array[1..30000] of Integer; (\*Mảng chứa các số ngẫu nhiên thế cho file\*)

Procedure Init; (\* Tao cac so ngau nhien cho X \*)

Var cs,t,i : integer;

Function RandInt(A,B:Integer):Integer;

(\* Tao so ngau nhien tren doan [A,B] \*)

Begin

RandInt:= A+ Random(B-A+1);

End;

begin

Randomize;

For i:=1 to So\_Max do X[i]:=i;

For i:=1 to 20000 do

begin cs:= RandInt(i,So\_Max);

t:=X[i];

X[i] := X[cs];

X[cs] :=t;

end;

end;

Procedure Sort;

Var i,csm, vt\_bit : integer;

Begin

For i:=1 to Sobyte do A[i]:=0;

For i:=1 to 20000 do

begin

csm:=( X[i] div 8) +1 ;

vt\_bit:= X[i] mod 8;

Case vt\_bit of

1: A[csm] := A[csm] or $80;

2: A[csm] := A[csm] or $40;

3: A[csm] := A[csm] or $20;

4: A[csm] := A[csm] or $10;

5: A[csm] := A[csm] or $08;

6: A[csm] := A[csm] or $04;

7: A[csm] := A[csm] or $02;

0: A[csm-1] := A[csm-1] or $01;

End;

end;

End;

Procedure Lietke;

Var i, j: Integer; mask,page: byte; c: char;

Begin page:=1;

gotoxy(1,1) ;writeln('Page ',page:3);

For i:=1 to Sobyte do

begin

mask:=$80;

For j:=1 to 8 do

begin

if (A[i] and mask) = mask then

write ((i-1)\*8+j:8);

if wherey = 24 then begin c:=readkey; clrscr;

inc(page);

gotoxy(1,1) ;writeln('Page ',page:3);

end;

mask := mask shr 1

end;

End;

End;

BEGIN

clrscr;

Init;

Sort; (\* Lấy dữ liệu từ X và bật bit tương ứng trong A lên 1 \*)

Lietke; (\* In dãy số trong A ra màn hình. \*)

readln;

END.