

Chương I

Bài 1.1: các hàm phủ định, hội, tuyển, kéo theo, tương đương mệnh đề

```
phudinhh := proc (p)  #Khai báo hàm  
    return 1-p;        #Trả về giá trị  
end proc:              #Kết thúc hàm
```

> phudinhh(0), phudinhh(1)

1, 0

```
hoi := proc (p, q)  
    if p = 1 and q = 1 then  
        return 1;  
    end if;  
    return 0;  
end proc:
```

> hoi(0, 0), hoi(0, 1), hoi(1, 0), hoi(1, 1)

0, 0, 0, 1

```
tuyen := proc (p, q)  
    if p = 0 and q = 0 then  
        return 0;  
    end if;  
    return 1;  
end proc:
```

> tuyen(0, 0), tuyen(0, 1), tuyen(1, 0), tuyen(1, 1)

0, 1, 1, 1

```
keotheo := proc (p, q)  
    if p = 1 and q = 0 then  
        return 0;  
    end if;  
    return 1;  
end proc:
```

Hoặc

```
keotheo := proc (p, q)  
    return tuyen(phudinhh(p), q);  
end proc:
```

> keotheo(0, 0), keotheo(0, 1), keotheo(1, 0), keotheo(1, 1)

1, 1, 0, 1

```
tuongduong := proc (p, q)  
    if p = q then
```

```
        return 1;
    end if;
    return 0;
end proc;
```

> tuongduong(0, 0), tuongduong(0, 1), tuongduong(1, 0), tuongduong(1, 1)

1, 0, 0, 1

Bài 1.2

```
chantri := proc ()
    local p, q, r, E;                #Khai báo biến toàn cục
    print(p, q, r, E);
    for p in [0, 1] do
        for q in [0, 1] do
            for r in [0, 1] do
                E := hoi(keothao(p, q), tuyen(phudinh(q), hoi(phudinh(q), r)));
                print(p, q, r, E);
            end do;                #Kết thúc for
        end do;
    end do;
end proc;
```

> chantri():

p, q, r, E
0, 0, 0, 1
0, 0, 1, 1
0, 1, 0, 0
0, 1, 1, 0
1, 0, 0, 0
1, 0, 1, 0
1, 1, 0, 0
1, 1, 1, 0

Bài 1.3

```
hangdung := proc ()
    local p, q, r, E, hd, hs;
    hd := true;
    hs := true;
    for p in [0, 1] do
        for q in [0, 1] do
            for r in [0, 1] do
                E := hoi(keothao(p, q), tuyen(phudinh(q), hoi(phudinh(q), r)));
                if E = 1 then
                    hs := false;
                else
                    hd := false;
                end if
            end do;
        end do;
    end do;
```

```
                end do;
            end do;
        end do;
        if hs then
            print("Menh de E la hang sai");
        elif hd then
            print("Menh de E la hang dung");
        else
            print("Menh de E khong phai hang dung cung khong phai hang sai");
        end if;
    end proc;
```

> hangdung()

"Menh de E khong phai hang dung cung khong phai hang sai"

Chương II

Bài 2.1: Kiểm tra phần tử x có thuộc tập hợp A hay không

```
ktthuoc := proc (x, A)           #Kiểm tra phần tử x có thuộc tập hợp A hay không.
    local y;                     #Khai báo biến cục bộ
    for y in A do
        if x = y then
            print("Phan tu x thuoc A");
            return;
        end if;
    end do;
    print("Phan tu x khong thuoc A");
end proc;
```

> ktthuoc(9, [1, 2, 5, 6, 7, 3, 1, 4])

"Phan tu x khong thuoc A"

> ktthuoc(5, [1, 2, 5, 6, 7, 3, 1, 4])

"Phan tu x thuoc A"

Bài 2.2: Hàm rút gọn một tập hợp

Ý tưởng: Cho một tập hợp B mới, với mỗi phần tử x trong A, nếu x chưa thuộc B thì đẩy x vào B. Kết quả là tập hợp B.

```
thuoc := proc (x, A)
    local y;
    for y in A do
        if x = y then
            return true;
        end if;
    end do;
```

```

    return false;
end proc:
rutgon := proc (A)
    local B, x;
    B := [];           #Khởi tạo B rỗng
    for x in A do
        if not thuộc(x, B) then
            B := [op(B), x];           #Nếu x chưa thuộc B, thì đây x vào vị trí cuối của B
        end if
    end do;
    return B;
end proc:

```

> rutgon([1, 2, 3, 2, 3])

[1, 2, 3]

Bài 2.3 (Đệ quy) Liệt kê các tập con của A

```

lietke := proc (A)
    local lk, B;
    B := [];           #Mảng B là mảng chứa kết quả mảng con của A
    lk := proc (A, i)   #Là hàm đệ quy liệt kê các mảng con của A xét tới phần tử thứ i-1
        if nops(A) < i then   #nops(A) là số phần tử của A
            print(B);         #Xuất ra mảng con B
            return;
        end if;
        lk(A, i+1);           #Gọi đệ quy, liệt kê các phần tử không có chứa A[i]
        B := [op(B), A[i]];    #Thêm A[i] vào mảng B
        lk(A, i+1);           #Gọi đệ quy, liệt kê các phần tử có chứa A[i]
        B := subsop(nops(B) = NULL, B); #Xóa A[i] ra khỏi B, trả về trạng thái ban đầu
    end proc;
    print("Cac tap con cua tap A la: ");
    lk(A, 1);
end proc:

```

> lietke([1, 2, 3, 4, 5]):

"Cac tap con cua tap A la: "

[]
 [5]
 [4]
 [4, 5]
 [3]
 [3, 5]
 [3, 4]
 [3, 4, 5]
 [2]
 [2, 5]

[2, 4]
[2, 4, 5]
[2, 3]
[2, 3, 5]
[2, 3, 4]
[2, 3, 4, 5]
[1]
[1, 5]
[1, 4]
[1, 4, 5]
[1, 3]
[1, 3, 5]
[1, 3, 4]
[1, 3, 4, 5]
[1, 2]
[1, 2, 5]
[1, 2, 4]
[1, 2, 4, 5]
[1, 2, 3]
[1, 2, 3, 5]
[1, 2, 3, 4]
[1, 2, 3, 4, 5]

Bài 2.4: Kiểm tra 2 tập hợp con và bằng nhau

```
con := proc (A, B)  #Ý tưởng: A là con B khi mọi phần tử của A đều thuộc B
    local x;
    for x in A do
        if not thuộc(x, B) then  #x thuộc A mà không thuộc B thì A không là con B
            return false;
        end if;
    end do;
    return true;
end proc;
bang := proc (A, B)  #A = B khi A là con B và ngược lại
    return con(A, B) and con(B, A);
end proc;
```

```
A := [1, 2, 3, 4, 5];
B := [1, 2, 3, 4, 5];
if bang(A, B) then
    print("A = B");
elif con(A, B) then
    print("A con B");
else
    print("A không là con B");
end if;
```

"A = B"

Bài 2.5: Các hàm hợp, giao, hiệu giữa 2 tập hợp

```

hop := proc (A, B)
    local x, C;
    C := rutgon(A); #C là tập hợp lưu kết quả. C= A Hợp B
    for x in B do
        if not thuộc(x, C) then #Với mỗi x thuộc B, nếu x chưa có trong C thì đẩy x vào C
            C := [op(C), x]; #Đẩy x vào vị trí cuối của C
        end if;
    end do;
    return C; #Hàm trả về kết quả là C
end proc;

giao := proc (A, B) #Ý tưởng: Với mỗi x thuộc A, nếu x thuộc B thì cho x vào tập kết quả
    local x, C;
    C := []; #Khởi tạo kết quả bằng rỗng
    for x in A do
        if thuộc(x, B) then
            C := [op(C), x]; #Đẩy x vào tập hợp C
        end if;
    end do;
    C := rutgon(C); #Rút gọn để tránh phần tử trùng nhau
    return C; #Trả kết quả
end proc;

hieu := proc (A, B) #Ý tưởng: với mỗi x thuộc A mà x không thuộc B, thì cho x vào tập kết quả C
    local x, C;
    C := [];
    for x in A do
        if thuộc(x, B) = false then
            C := [op(C), x];
        end if;
    end do;
    C := rutgon(C);
    return C;
end proc;

```

> hop([1, 2, 3], [3, 2, 5])

[1, 2, 3, 5]

Bài 2.6

```

E := [1, 2, 3, 4, 5];
A := [1, 3, 5];
print("Phan bu cua A la : ", hieu(E, A));

```

"Phan bu cua A la : ", [2, 4]

Bài 2.7

```

donanh := proc (a)

```

```
local b;  
b := rutgon(a);  
if nops(a) = nops(b) then  
    print("Anh xa do la don anh");  
else  
    print("Anh xa do khong phai la don anh");  
end if;  
end proc:
```

> donanh([1, 2, 3, 4, 6]);

"Anh xa do la don anh"

> donanh([1, 2, 1, 10, 3]);

"Anh xa do khong phai la don anh"

```
songanh := proc (a)  
    local x, b;  
    for x to nops(a) do  
        if not thuoc(x, a) then  
            print("Day khong phai la song anh");  
            return;  
        end if;  
    end do;  
    print("Day la song anh");  
    print("Anh xa nguoc cua no la ");  
    b := [seq(0, i = 1 .. nops(a))];  
    for x to nops(a) do  
        b[a[x]] := x  
    end do;  
    print(b);  
end proc:
```

> songanh([5, 1, 2, 4, 3]);

"Day la song anh"
"Anh xa nguoc cua no la "
[2, 3, 5, 4, 1]

```
lkdonanh := proc (n, m)  
    local lk, A;  
    A := [seq(0, i = 1 .. n)];  
    lk := proc (fill, i)  
        local x;  
        if fill < n-i+1 then  
            return;  
        end if;  
        if n < i then  
            print(A);  
            return;  
        end if;  
    end proc;  
    lkdonanh(n, m, fill, i);  
end proc;
```

```
        end if;
        for x to m do
            if not (x in A) then
                A[i] := x;
                lk(fill-1, i+1);
                A[i] := 0;
            end if;
        end do;
    end proc;
    lk(m, 1);
end proc;
```

```
lktoananh := proc (n, m)
    local lk, A;
    A := [seq(0, i = 1 .. n)];
    lk := proc (fill, i)
        local x;
        if n < fill+i-1 then
            return;
        end if;
        if n < i then
            print(A);
            return;
        end if;
        for x to m do
            if x in A then
                if fill+i-1 < n then
                    A[i] := x;
                    lk(fill, i+1);
                    A[i] := 0;
                end if;
            else
                A[i] := x;
                lk(fill-1, i+1);
                A[i] := 0;
            end if;
        end do;
    end proc;
    lk(m, 1);
end proc;
```

```
n := 3;
m := 3;
print("Nhưng đơn anh X vào Y");
lkdonanh(n, m);
print("Nhưng toán anh X vào Y");
```


lktoananh(n, m):

"Nhưng don anh X vào Y"

[1, 2, 3]

[1, 3, 2]

[2, 1, 3]

[2, 3, 1]

[3, 1, 2]

[3, 2, 1]

"Nhưng toan anh X vào Y"

[1, 2, 3]

[1, 3, 2]

[2, 1, 3]

[2, 3, 1]

[3, 1, 2]

[3, 2, 1]

Chương III

Bài 3.1 Viết chương trình liệt kê hết tất các hoán vị của tập hợp $\{1, 2, 3, \dots, n\}$ với

Input: $n > 0$ và Output là danh sách các hoán vị của $\{1, 2, \dots, n\}$

NextHoanVi := **proc** (Y) **#Ý tưởng: Các bạn tìm đọc thêm thuật toán sinh hoán vị để hiểu rõ**

local i, j, X;

X := Y; **#Biến tạm X:=Y, vì maple không cho thay đổi giá trị tham trị**

i := nops(X); **#i=n, nops(X) là số phần tử của X**

while 1 < i **and** X[i] <= X[i-1] **do** **#Tìm đoạn giảm dần dài nhất**

i := i-1;

end do;

if i <= 1 **then** **#Đã đến hoán vị cuối cùng thì thoát**

return false;

end if;

j := nops(X);

while i < j **and** X[j] < X[i-1] **do** **#Tìm phần tử đầu tiên sao cho X[j] > X[i-1]**

j := j-1;

end do;

X[i-1], X[j] := X[j], X[i-1]; **#Đổi chỗ X[j] và X[i-1]**

j := nops(X); **#Lật đoạn từ i -> n**

while i < j **do**

X[i], X[j] := X[j], X[i];

i := i+1;

j := j-1;

end do;

return X; **#Trả về cấu hình kế tiếp**

end proc;

Hoanvi := **proc** (n)

```

local X;
X := [seq(i, i = 1 .. n)];           #Khởi tạo X là hoán vị đầu tiên [1,2,3,...,n]
while X <> false do                 #Khi nào còn sinh được thì con làm
    print(X);                        #In hoán vị
    X := NextHoanVi(X);              #Tạo ra hoán vị kế tiếp
end do;
end proc;

```

> Hoanvi(4);

```

[1, 2, 3, 4]
[1, 2, 4, 3]
[1, 3, 2, 4]
[1, 3, 4, 2]
[1, 4, 2, 3]
[1, 4, 3, 2]
[2, 1, 3, 4]
[2, 1, 4, 3]
[2, 3, 1, 4]
[2, 3, 4, 1]
[2, 4, 1, 3]
[2, 4, 3, 1]
[3, 1, 2, 4]
[3, 1, 4, 2]
[3, 2, 1, 4]
[3, 2, 4, 1]
[3, 4, 1, 2]
[3, 4, 2, 1]
[4, 1, 2, 3]
[4, 1, 3, 2]
[4, 2, 1, 3]
[4, 2, 3, 1]
[4, 3, 1, 2]
[4, 3, 2, 1]

```

Bài 2. Viết chương trình liệt kê hết tất các chỉnh hợp chập k của tập hợp $\{1,2,3,\dots,n\}$ với Input: $n>0$ và $k\leq n$ Output là danh sách các chỉnh hợp (Không theo thứ tự từ điển)

```

NextToHop := proc (Y, m)             #Tham khảo thuật toán sinh tổ hợp
    local i, j, X, n;
    X := Y;
    n := m;
    i := nops(X);
    while 0 < i and X[i] = n do
        i := i-1;
        n := n-1;
    end do;
    if i = 0 then
        return false;
    end if;

```

```
    end if;
    X[i] := X[i]+1;
    for i from i+1 to nops(X) do
        X[i] := X[i-1]+1;
    end do;
    return X;
end proc;

Chinhhop := proc (n, k)           #Ý tưởng: Sinh các tổ hợp rồi hoán vị các tổ hợp
    local X, Y;
    Y := [seq(i, i = 1 .. k)];
    while Y <> false do
        X := Y;
        while X <> false do
            print(X);
            X := NextHoanVi(X);
        end do;
        Y := NextToHop(Y, n);
    end do;
end proc;
```

> Chinhhop(5, 3);

[1, 2, 3]
[1, 3, 2]
[2, 1, 3]
[2, 3, 1]
[3, 1, 2]
[3, 2, 1]
[1, 2, 4]
[1, 4, 2]
[2, 1, 4]
[2, 4, 1]
[4, 1, 2]
[4, 2, 1]
[1, 2, 5]
[1, 5, 2]
[2, 1, 5]
[2, 5, 1]
[5, 1, 2]
[5, 2, 1]
[1, 3, 4]
[1, 4, 3]
[3, 1, 4]
[3, 4, 1]
[4, 1, 3]
[4, 3, 1]
[1, 3, 5]

[1, 5, 3]
[3, 1, 5]
[3, 5, 1]
[5, 1, 3]
[5, 3, 1]
[1, 4, 5]
[1, 5, 4]
[4, 1, 5]
[4, 5, 1]
[5, 1, 4]
[5, 4, 1]
[2, 3, 4]
[2, 4, 3]
[3, 2, 4]
[3, 4, 2]
[4, 2, 3]
[4, 3, 2]
[2, 3, 5]
[2, 5, 3]
[3, 2, 5]
[3, 5, 2]
[5, 2, 3]
[5, 3, 2]
[2, 4, 5]
[2, 5, 4]
[4, 2, 5]
[4, 5, 2]
[5, 2, 4]
[5, 4, 2]
[3, 4, 5]
[3, 5, 4]
[4, 3, 5]
[4, 5, 3]
[5, 3, 4]
[5, 4, 3]

Bài 2 (Đệ quy theo thứ tự từ điển)

```
Chinhhop2 := proc (n, k)
    local a, lietke;
    a := [seq(0, i = 1 .. k)];           #Khởi tạo mảng a gồm k phần tử có giá trị 0
    lietke := proc (i)                   #Là hàm đệ quy để liệt kê chỉnh hợp tại vị trí thứ i
        local j, t;
        if k < i then print(a);         #Nếu i>k, là kết thúc cấu hình chỉnh hợp, in kết quả, thoát
            return;
        end if;
        for j to n do
            if not (j in a) then         #Nếu j chưa thuộc a thì gán a[i]=j
                a[i] := j;
            end if;
        end for;
    end proc;
end proc;
```

<pre> lietke(i+1); a[i] := 0; end if; end do; end proc; lietke(1); end proc: </pre>	<pre> #Tiếp tục làm ở vị trí thứ i+1 #Trả về trạng thái ban đầu </pre>
---	--

> Chinhhop2(5, 3);

```

[1, 2, 3]
[1, 2, 4]
[1, 2, 5]
[1, 3, 2]
[1, 3, 4]
[1, 3, 5]
[1, 4, 2]
[1, 4, 3]
[1, 4, 5]
[1, 5, 2]
[1, 5, 3]
[1, 5, 4]
[2, 1, 3]
[2, 1, 4]
[2, 1, 5]
[2, 3, 1]
[2, 3, 4]
[2, 3, 5]
[2, 4, 1]
[2, 4, 3]
[2, 4, 5]
[2, 5, 1]
[2, 5, 3]
[2, 5, 4]
[3, 1, 2]
[3, 1, 4]
[3, 1, 5]
[3, 2, 1]
[3, 2, 4]
[3, 2, 5]
[3, 4, 1]
[3, 4, 2]
[3, 4, 5]
[3, 5, 1]
[3, 5, 2]
[3, 5, 4]
[4, 1, 2]
[4, 1, 3]

```

[4, 1, 5]
[4, 2, 1]
[4, 2, 3]
[4, 2, 5]
[4, 3, 1]
[4, 3, 2]
[4, 3, 5]
[4, 5, 1]
[4, 5, 2]
[4, 5, 3]
[5, 1, 2]
[5, 1, 3]
[5, 1, 4]
[5, 2, 1]
[5, 2, 3]
[5, 2, 4]
[5, 3, 1]
[5, 3, 2]
[5, 3, 4]
[5, 4, 1]
[5, 4, 2]
[5, 4, 3]

Bài 3. Viết chương trình tính nghiệm nguyên không âm của phương trình $x_1 + \dots + x_k = n$ với Input: n và k Output là danh sách các bộ nghiệm

```
NextNghiem := proc (Y)
    local i, j, X, t;
    X := Y;
    i := nops(X)-1;
    while 0 < i and X[i] = 0 do #Tìm vị trí có thể còn giảm được, nhưng không phải vị trí cuối
        i := i-1;
    end do;
    if i <= 0 then
        return false;
    end if;
    X[i] := X[i]-1;           #Giảm vị trí xuống
    t := X[nops(X)]+1;        #Do tổng = n, nên t phải bằng vị trí cuối cộng 1
    X[nops(X)] := 0;          #Đổi chỗ vị trí cuối và i+1
    X[i+1] := t;
    return X;
end proc;

Nghiem := proc (n, k)
    local Y;
    Y := [seq(0, i = 1 .. k)];
    Y[1] := n;                #Khởi tạo cấu hình cuối là (n,0,0,...)
    while Y <> false do
        print(Y);             #In ra kết quả
    end while;
end proc;
```

```
Y := NextNghiem(Y);    #Tìm cấu hình tiếp theo
end do;
end proc;
```

Bài 4. Viết chương trình liệt kê hết tất các chuỗi bit có độ dài n và chứa đúng k bit 1 với Input: n>0 và k≤n Output là danh sách các chuỗi bit.

```
Chuoibit := proc (n, k) #Ý tưởng: Sinh tổ hợp là k vị trí trong chuỗi bit bằng 1
local X, x, i;
X := [seq(i, i = 1 .. k)];
while X <> false do
x := [seq(0, i = 1 .. n)];
for i to k do
x[X[i]] := 1;    #Gán bit tại vị trí X[i] bằng 1
end do;
print(cat(op(x)));
X := NextToHop(X, n);
end do;
end proc;
```

> Chuoibit(6, 3);

```
111000
110100
110010
110001
101100
101010
101001
100110
100101
100011
011100
011010
011001
010110
010101
010011
001110
001101
001011
000111
```

Chương IV

Bài 4.1: Cho n là số nguyên dương. Viết chương trình tính phần tử f_n của dãy Fibonacci

```
fibonacci := proc (n)
local a, b, c, i;
a := 1;
```

```
b := 1;
if n = 1 then
    return a;
end if;
if n = 2 then
    return b;
end if;
for i from 3 to n do
    c := a+b;
    a := b;
    b := c
end do;
return c;
end proc;
```

> fibonacci(1000);

43466557686937456435688527675040625802564660517371780402481729089536555417949051894
03879840079255169295922593080322634775209689623239873322471161642996440906533187938
298969649928516003704476137795166849228875

Bài 4.2: Xét bài toán tháp Hà Nội, gồm 3 cọc A, B, C và n đĩa ở cọc A. Hãy viết chương trình liệt kê các bước di chuyển từ cọc A sang cọc C

```
thapn := proc (n, x, y, z)
    if n = 0 then
        return;
    end if;
    thapn(n-1, x, z, y);
    print(cat("Di chuyen tu ", x, " sang ", z));
    thapn(n-1, y, x, z);
end proc;
```

> thapn(3, "A", "B", "C");

"Di chuyen tu A sang C"
"Di chuyen tu A sang B"
"Di chuyen tu C sang B"
"Di chuyen tu A sang C"
"Di chuyen tu B sang A"
"Di chuyen tu B sang C"
"Di chuyen tu A sang C"

Bài 4.3: Cho hệ thức đệ quy $x_n = a x_{(n-1)} + b x_{(n-2)}$, với a, b thuộc R và điều kiện đầu $a_0 = C_0$ và $a_1 = C_1$. Tính a_k

```
tinhhtdq := proc (a, b, C0, C1, k)
    local x, y, z, i;
    if k = 0 then return C0; end if;
    if k = 1 then return C1; end if;
    x := C0; y := C1;
```



```

    for i from 2 to k do
        z := a*y+b*x;
        x := y;
        y := z;
    end do;
    return z;
end proc;

```

> tinhhtdq(1.4, .25, 0, 1, 100);

1.211398130 10¹⁹

Bài 4.4: Cho hệ thức đệ quy $f(n) = a f(n/m) + b$, $f(1) = c$ với a, b, c thuộc \mathbb{R} và m là số nguyên dương. Hãy viết chương trình tính $f(m^k)$ với k là số nguyên dương.

```

tinhf := proc (a, b, c, k)
    local f, i;
    if k = 0 then return c; end if;
    f := c;
    for i to k do
        f := f*a+b;
    end do;
    return f;
end proc;

```

> tinhf(5, 2, 19, 199);

24269759583658452657861849809742484538301983458012551819094354797679825452737995420
6173848101902831061321696992649776802863925695419311523437

Chương V

Bài 5.1. Cho n và b là hai số nguyên dương lớn hơn 1. Hãy viết chương trình để tìm biểu diễn của n theo cơ số b

```

chuyencosob := proc (n, b)
    local x, m;
    m := n;
    x := [];
    while 0 < m do
        x := [irem(m, b), op(x)];
        m := iquo(m, b);
    end do;
    return x;
end proc;

```

#m là biến tạm của n
#x là mảng kết quả, khởi tạo bằng rỗng
#lấy m mod b, đưa vào vị trí đầu tiên của mảng x
gán m = m div b
#x là mảng kết quả

> chuyencosob(16, 2);

[1, 0, 0, 0, 0]

Bài 5.2 Cho n và m được biểu diễn dưới dạng cơ số b . Hãy viết chương trình tính tổng và tích và biểu diễn chúng dưới dạng cơ số b

```
chuyenthanhso := proc (n, b)           #Chuyển một số ở cơ số b về hệ thập phân
    local i, x;
    x := 0;
    for i to nops(n) do
        x := x*b+n[i];
    end do;
end proc;
```

> chuyenthanhso([1, 2, 3, 4, 5], 6);

1865

```
tong := proc (n, m, b)
    return chuyencosob(chuyenthanhso(n, b)+chuyenthanhso(m, b), b);
end proc;
tich := proc (n, m, b)
    return chuyencosob(chuyenthanhso(n, b)*chuyenthanhso(m, b), b);
end proc;
```

> tong([1, 0, 1, 1, 1], [1, 1, 0, 1, 1, 0], 2);

[1, 0, 0, 1, 1, 0, 1]

> tich([1, 1, 0, 1, 0], [0, 1, 1, 0, 0], 2);

[1, 0, 0, 1, 1, 1, 0, 0, 0]

Bài 5.3 Cho b, c là số nguyên dương lớn hơn 1 và một biểu diễn n theo cơ số b . Hãy viết chương trình để tìm biểu diễn của n theo cơ số c .

```
chuyencoso := proc (b, c, n)
    return chuyencosob(chuyenthanhso(n, b), c);
end proc;
```

> chuyencoso(8, 10, [5, 2, 4, 1]);

[2, 7, 2, 1]

Bài 5.4 Cho a, b là hai số nguyên dương. Viết chương trình để tính ước chung lớn nhất của a và b theo thuật toán Euclid

```
ucln := proc (a, b)
    local x, y, t;
    x := a; y := b;
    while y <> 0 do
        t := irem(x, y);
        x := y;
        y := t;
    end do;
    return x;
```

end proc;

> ucln(9, 6);

3

Bài 5.5 Cho a, b là hai số tự nhiên. Gọi $d=(a,b)$, hãy viết chương trình để tìm m, n sao cho $d = m a + n b$

```
timmn := proc (a, b)           #Tham khảo ý tưởng thuật toán Euclide mở rộng
    local x, y, t, m, n, u, v, p, q;
    x := a; y := b;
    m := 1; n := 0;
    u := 0; v := 1;
    while y <> 0 do
        q := iquo(x, y);
        t := x-y*q;
        x := y;
        y := t;
        p := m-u*q;
        q := n-v*q;
        m, n := u, v;
        u, v := p, q;
    end do;
    return m, n;
end proc;
```

> timmn(17, 3);

-1, 6