LẬP TRÌNH C CĂN BẢN

Th.S: Dương Thị Thùy Vân

Khoa CNTT & TƯD

CHƯƠNG 4

PHÉP TOÁN VÀ BIỂU THỨC

Nội dung

1. Khái niệm biểu thức
2. Phép toán
3. Phép toán số học
4. Phép toán quan hệ
5. Phép toán luận lý
6. Chuyển kiểu
7. Tăng và giảm
8. Phép gán và biểu thức gán
9. Thứ tự thực hiện phép toán

1. Khái niệm biểu thức

- Là sự kết hợp hợp lệ giữa các toán hạng và toán tử để diễn đạt một công thức toán học nào đó, cho một kết quả duy nhất sau cùng.

  Ví dụ:
  ```
  delta= b*b - 4*a*c;
  pi= 4*atan(1.0);
  ```

- Biểu thức với toán tử là phép toán số học → ***biểu thức số* học**
- Với phép toán quan hệ & luận lí → ***biểu thức quan hệ & luận lí.***

2. Phép toán

- Trong C, các phép toán có thể phân ra thành 3 loại chính: phép toán số học, phép thao tác bit, phép toán quan hệ và luận lý.
- Phép toán 1 ngôi, còn gọi là phép toán 1 toán hạng.
- Phép toán 2 ngôi, còn gọi là phép toán 2 toán hạng.
- Độ ưu tiên của phép toán qui định trình tự tính toán trong biểu thức.

Ví dụ:

```
a = - 9/2*2 - 2 - 7%5;
```

3. Phép toán số học

- Các phép toán số học 1 ngôi: **+ -**
- Các phép toán số học 2 ngôi: *** / % + -**
- Phép chia nguyên và chia không nguyên: /

  Ví dụ: 11/2 = 5 11/2.0 = 5.5
- Phép toán % cho phần dư của phép chia nguyên.
- Phép toán % không áp dụng được cho các giá trị kiểu float và double.

4. Phép toán quan hệ

- Phép toán quan hệ: > < <= >=
  == !=
- Phép toán quan hệ cho ta hoặc giá trị đúng (1) hoặc giá trị sai (0).

  Ví dụ:

```
if (a>b)
        cout<<a<<" la so lon hon !";
if (b!=0)
        cout<<a/b;
```

- Các phép toán quan hệ có độ ưu tiên thấp hơn so với các phép toán số học.

5. Phép toán luận lí

- Phép toán luận lí: **&&** **||** **!**
  and ***or*** ***not***
- Phép toán luận lý cho ta hoặc giá trị đúng (1) hoặc giá trị sai (0).

  Ví dụ: 3 && 7 có giá trị 1
- Các phép toán quan hệ và luận lí được dùng để thiết lập điều kiện rẽ nhánh trong toán tử if và điều kiện kết thúc chu trình trong các toán tử for, while và do-while.

6. Chuyển kiểu (1)

- Trong một biểu thức, các toán hạng khác kiểu sẽ phải chuyển sang cùng kiểu để tính toán.
- Chuyển kiểu *tự động* và chuyển kiểu *tường minh*.
  - (1) Việc ***tự động*** chuyển kiểu được thực hiện từ toán hạng có kiểu "hẹp" sang kiểu "rộng" hơn.

Ví dụ:

```
x = - 9.0/4*2/2 - 2 - 7%5;
y = - 9.0/4*2%2 - 2 - 7%5; //??
```

6. Chuyển kiểu (2)

- Với phép gán, kết quả của biểu thức bên phải sẽ được chuyển thành *kiểu của biến* bên trái.

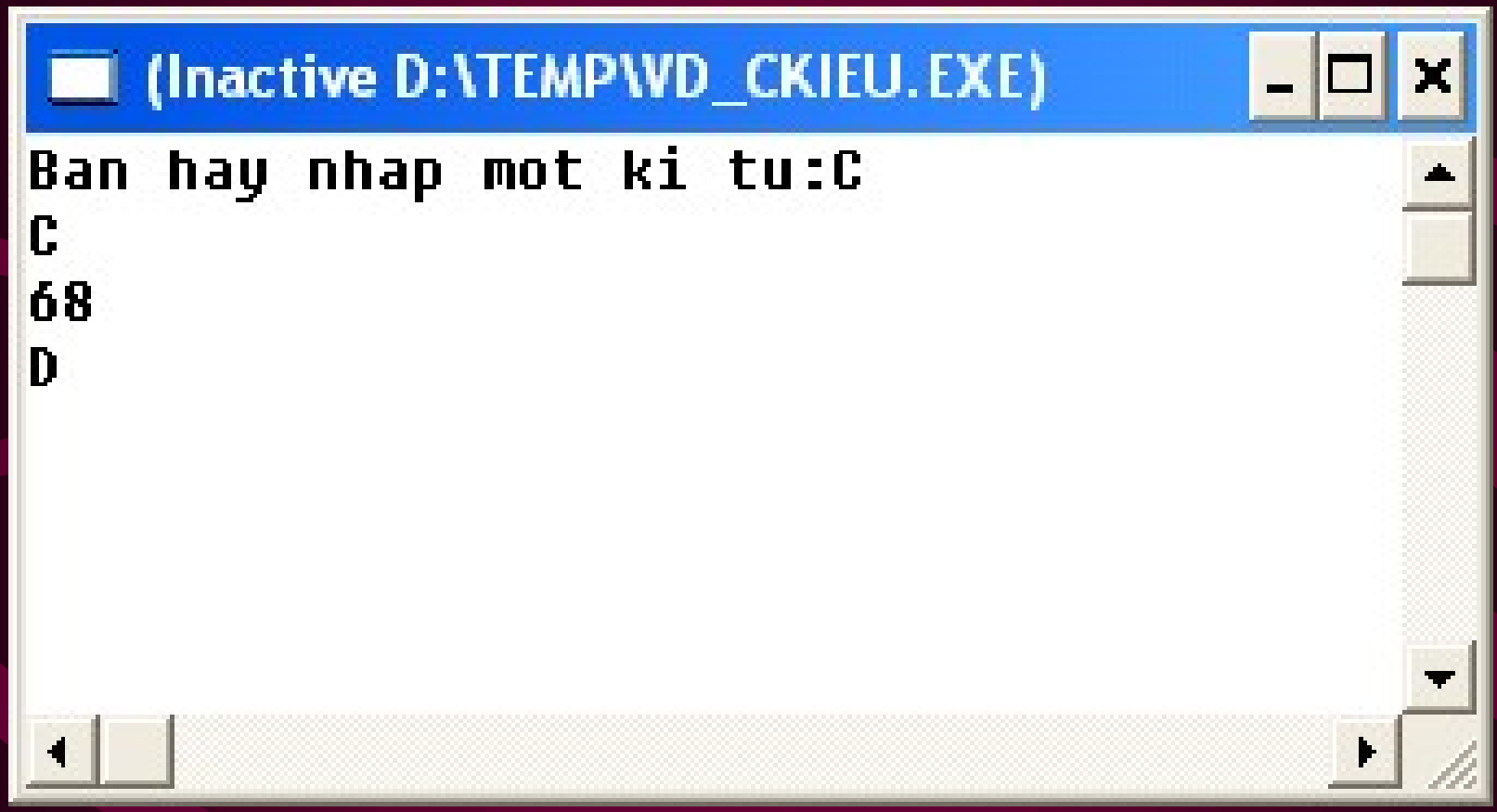
Ví dụ:

```
float x;
x = 3/4.0 + 2;
int y;
y = 3/4.0 + 2;
```

d:\temp\vd_ckieu.cpp

```
#include<iostream.h>
void main()
{
char t;
cout<<"Ban hay nhap mot ki tu:";
cin>>t;
cout<<t   <<'\n';
cout<<t+1 <<'\n';
t= t+1;
cout<<t;
}
```

Kết quả



6. Chuyển kiểu (3)

(2) Chuyển kiểu ***tường minh***:

Buộc kiểu của biểu thức chuyển sang kiểu khác.

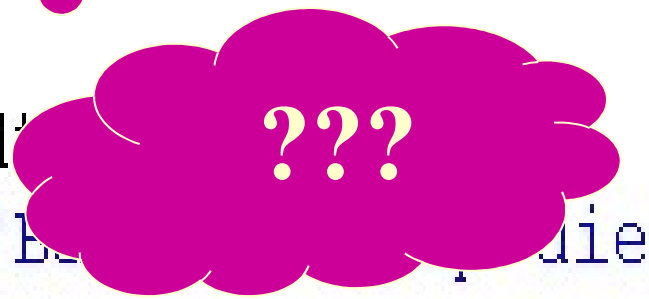
`(KDL)BTh` `KDL(BTh)`

Ví dụ:

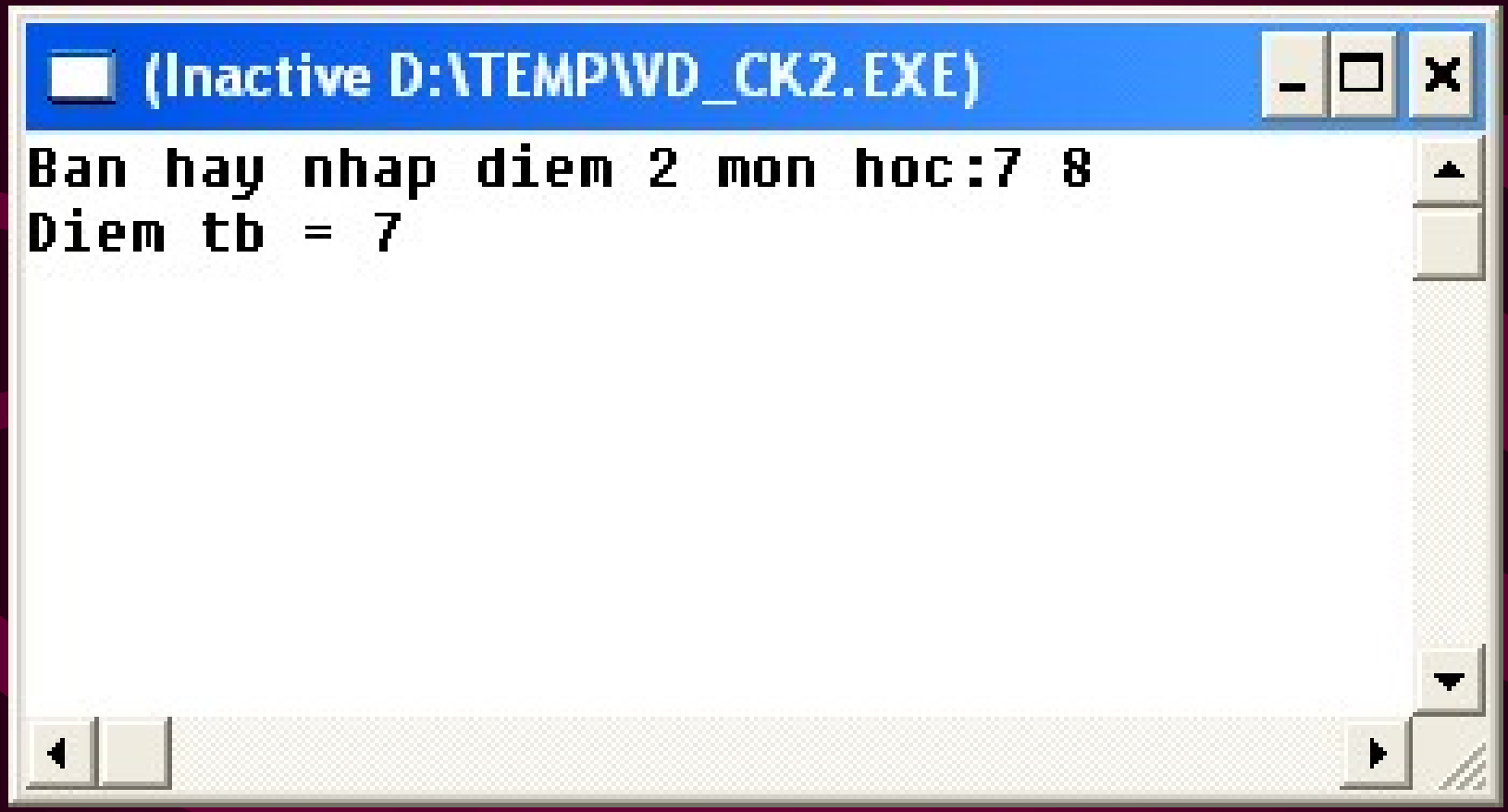
```
long a= 300000 + (long)400000;
double x= double(3)/4*4.0f;
double y= double(1/2)*100; //??
long s= s + long(n)*17000;
```

d:\temp\vd_ck2.cpp

```
#include<iostream.h>
#include<iomanip.h>
const int dvht1= 8, dvht2= 10;
void main()
{
    int d1, d2;
        float d
        cout<<"B          diem 2 mon hoc:";
    cin>>d1>>d2;
    dtb = (dvht1*d1 + dvht2*d2)/(dvht1+dvht2);
    cout<<"Diem tb = "<<setprecision(2)<<dtb;
}
```



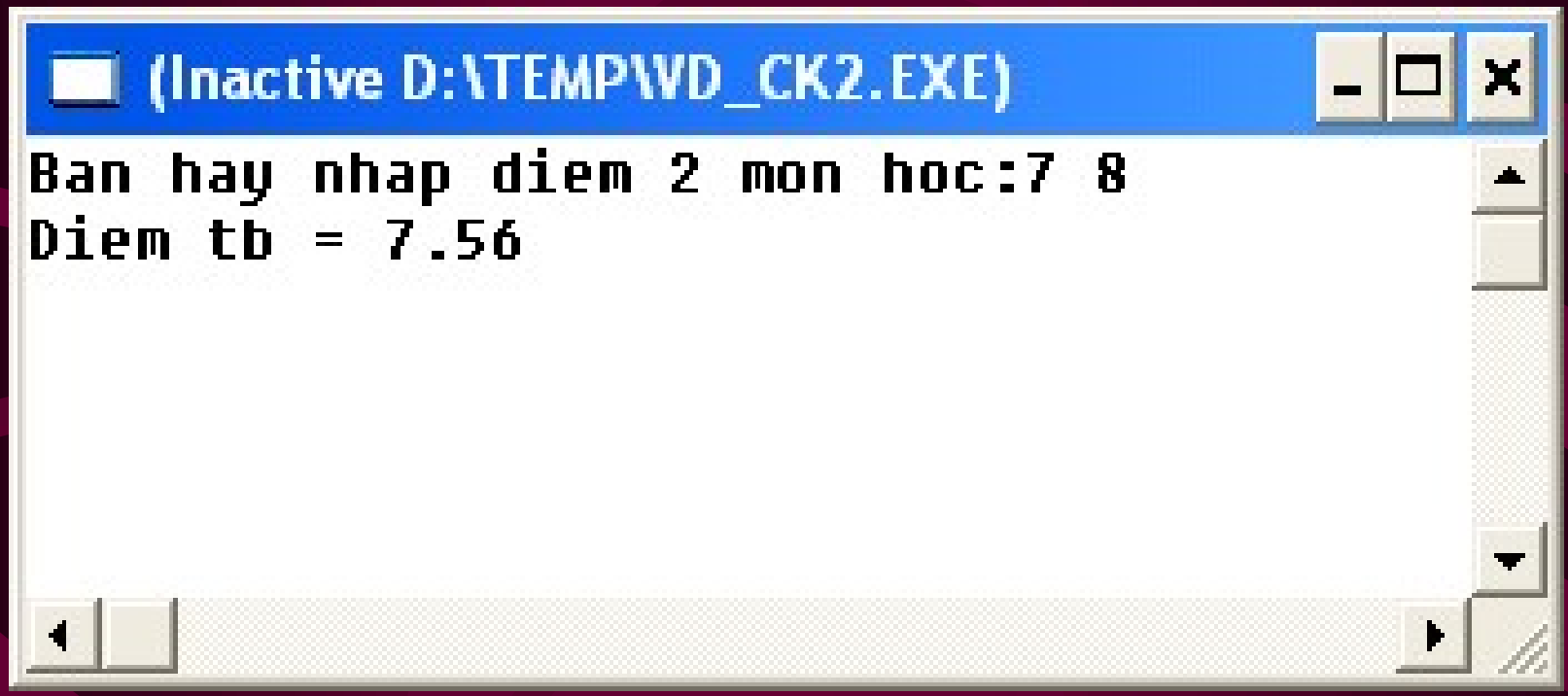
Kết quả


(Inactive D:\TEMP\VD_CK2.EXE)
Ban hay nhap diem 2 mon hoc:7 8
Diem tb = 7

d:\temp\vd_ck2.cpp

```
#include<iostream.h>
#include<iomanip.h>
const int dvht1= 8, dvht2= 10;
void main()
{
    int d1, d2;
    float dtb;
    cout<<"Ban hay nhap diem 2 mon hoc:";
    cin>>d1>>d2;
    dtb = float(dvht1*d1 + dvht2*d2)/(dvht1+dvht2);
    cout<<"Diem tb = "<<setprecision(2)<<dtb;
}
```

Kết quả


(Inactive D:\TEMP\VD_CK2.EXE)
Ban hay nhap diem 2 mon hoc:7 8
Diem tb = 7.56

Kiểu bool (1)

- Kiểu ***bool*** được dùng để biểu diễn kết quả của biểu thức luận lí, cho kết quả là ***đúng*** (***true***) hoặc ***sai*** (***false***).
- Ngôn ngữ C không định nghĩa tường minh kiểu bool, được dùng thông qua *kiểu số nguyên*.
  - Kết quả biểu thức là ***true*** $\Rightarrow$ giá trị là ***1***
  - Kết quả biểu thức là ***false*** $\Rightarrow$ giá trị là ***0***

Kiểu bool (2)

```
int a, b, c;
cin>>a>>b;
c= a>b;      //c= 0 or 1
```

- Giá trị biểu thức là $\neq 0 \Rightarrow$ KQ ứng là ***true***
- Giá trị biểu thức là $= 0 \Rightarrow$ KQ ứng là ***false***

```
if (b)
    cout<<a/b;
```

Ví dụ 1

Kiểm tra một năm ***y*** có phải là năm nhuận ? Biết n*ăm là nhuận nếu là năm chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100.*

```
int y;
cout<<"Ban hay nhap mot nam: ";
cin>>y;
if ((y%4==0 && y%100!=0)||y%400 == 0)
    cout<<y<<" la nam nhuan !";
else
    cout<<y<<" khong la nam nhuan !";
```

Ví dụ 2

Kiểm tra a, b, c có thể là 3 cạnh của một tam giác ? *Tổng chiều dài của hai cạnh bất kì luôn lớn hơn chiều dài cạnh còn lại.*

```
int a, b, c;
cout<<"Ban hay nhap 3 so nguyen: ";
cin>>a>>b>>c;
if ( a+b>c && a+c>b && c+b>a )
     cout<<"Thoa 3 canh mot tam giac!";
else
     cout<<"Khong thoa ... ";
```

⇒ Làm lại cách khác dùng mệnh đề và phép phủ định !!

Ví dụ 3

Tính tổng S = 1+3+5+...n ? (n ≥ 0)

```
int n, S = 0;
cout<<"Ban hay nhap so nguyen duong: ";
cin>>n;
for (int i= 1; i<n; i=i+2)
     S = S+i;
cout<<"Tong so le nho hon n: S ="<<S;
```

Toán tử sizeof

- Cho biết kích thước (theo byte) của kiểu dữ kiệu cơ sở (hoặc của một đối tượng cụ thể).

Ví dụ:

```
int n= sizeof(long);    //n= 4
1= sizeof(char) ≤ sizeof(short) ≤
      <= sizeof(int) ≤ sizeof(long)
```

7. Phép tăng và giảm (1)

- Nếu phép ***tăng*** (ký hiệu ++) đặt ngay *trước tên biến*, là phép toán "*tăng trước*".
- Tương tự, phép ***giảm*** (ký hiệu --) đặt ngay *trước tên biến*, là phép toán "*giảm trước*".
- Giá trị của biến được *tăng (giảm)* 1 đơn vị, sau đó giá trị mới này được ***dùng trong biểu thức*** mà biến xuất hiện.

```
int a= 5, b= 6, c;
c= ++a + b; c= a * --b;
```

7. Phép tăng và giảm (2)

- Nếu phép ***tăng*** đặt ngay *sau tên biến*, là phép toán "*tăng sau*".
- Nếu phép ***giảm*** đặt ngay *sau tên biến*, là phép toán "*giảm sau*".
- Giá trị hiện tại của biến được ***dùng trong biểu thức*** mà nó xuất hiện, sau đó giá trị của biến mới được ***tăng (giảm) 1*** đơn vị.

```
int a= 5, b= 6, c;
c= a++ + b; c= a * b--;
```

noname00.cpp

```
//Chuong trình tính tong cac so tu nhien nho hon N
#include<iostream.h>
void main()
{
    int n, S = 0;
    cout<<"Moi nhap vap so duong n: ";
    cin>>n;
    for (int i= 1; i<=n; i++)
        S = S+i;
    cout<<"Tong cac so tu nhien nho hon "<<n<< " la: "<< S;
}
```

8. Phép gán và biểu thức gán (1)

- Phép gán "=" để thay đổi giá trị biến:

  tenBien = giatri;

- Chú ý phân biệt toán tử gán với khái niệm đẳng thức trong toán học.
- Biểu thức gán là biểu thức có dạng:

  v = e

- Trong đó v là một biến, e là một biểu thức.Giá trị của biểu thức gán là giá trị của e, kiểu của nó là kiểu của v.

8. Phép gán và biểu thức gán (2)

- Nếu đặt dấu ; vào sau biểu thức gán thì ta được toán tử gán:

  v = e;

- Biểu thức gán có thể sử dụng trong các phép toán và các câu lệnh như các biểu thức khác. Ví dụ 1:

  a = b = 5;

thì điều đó có nghĩa là gán giá trị của biểu thức:

  b = 5

cho biến a. Kết quả là b = 5 và a = 5.

- Ví dụ 2: Sau khi thực hiện câu lệnh

  z = (y = 2)*(x = 6);

thì y có giá trị 2, x có giá trị 6 và z có giá trị 12.

8. Phép gán và biểu thức gán (3)

- Phép toán kết hợp là phép gán cùng với phép toán, tác động lên chính biến được gán.

  += -= *= /= %=

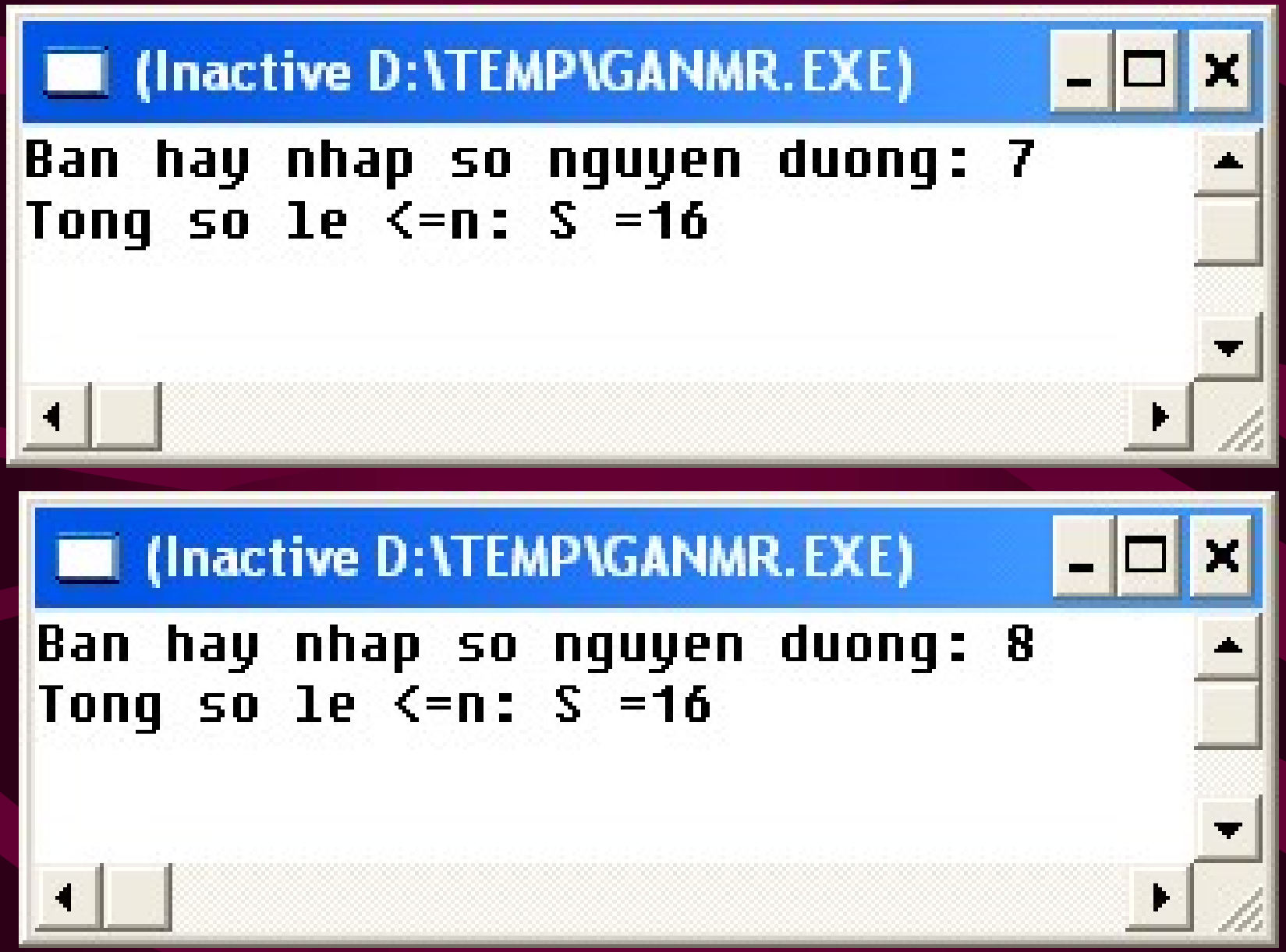
  &= |= ^= <<= >>=

Ví dụ:

```
i += 2;          //⇔ i = i + 2;
a *= b+1         //⇔ a = a*(b + 1);
a <<= 1          //⇔ a = a<<1;
```

d:\temp\ganmr.cpp

```
/*Chuong trinh tinh tong cac so le
     S= 1 + 3 + ... + <=n     */
#include<iostream.h>
void main()
{
     int n, S = 0;
     cout<<"Ban hay nhap so nguyen duong: ";
     cin>>n;
     for (int i= 1; i<=n; i+=2)
          S += i;
     cout<<"Tong so le <=n: S ="<<S;
}
```


(Inactive D:\TEMP\GANMR.EXE)
Ban hay nhap so nguyen duong: 7
Tong so le <=n: S =16
(Inactive D:\TEMP\GANMR.EXE)
Ban hay nhap so nguyen duong: 8
Tong so le <=n: S =16

9. Thứ tự thực hiện phép toán

- Khi thực hiện tính toán trong một biểu thức, phép toán có độ ưu tiên cao hơn sẽ thực hiện trước.

```
b= (a= 3)+2;
b= a= 3 + 2;
n= 18/4*4;
```

- Bảng sau cho biết *độ ưu tiên* phép toán (thứ tự giảm dần).
- Trừ phép gán và phép 1 toán hạng, các phép cùng cấp sẽ ưu tiên trái hơn.

Ưu tiên giảm dần	Ngoặc đơn	`( )`	
	Phép một toán hạng (←)	`! ~ ++ -- + - (type) sizeof`	
	Cùng cấp phép nhân	`* / %`	
	Cùng cấp phép cộng	`+ -`	`>> <<`
	Phép toán quan hệ	`< <= > >=`	`== !=`
	Phép toán luận lí	`& | ^`	`&& ||`
	Phép gán (←)	`= += -= *= /= %= &= |= ^= <<= >>=`	

Ví dụ

```
(3.0/4 < 4.0/5) && (‘a’ < ‘b’)
(3/4 < 4/5) && (‘a’ < ‘b’)
!(48.5+2 < 50) || (2 > 4/2)
!(48.5+2 < 50) && (3> 4/2)
n&1==0
```

Bài tập

- Giá trị của x là 10, x và a là bao nhiêu sau khi thực thi:

```
a = x++;
```

- Giá trị của x là 10, x và a là bao nhiêu sau khi thực thi:

```
a = ++x;
```