

CHƯƠNG 3

BIẾN VÀ HẲNG

ThS. Dương Thị Thùy Vân

Khoa CNTT & TỬĐ

Nội dung

1. Danh hiệu
2. Từ khóa
3. Kiểu dữ liệu
4. Khái niệm biến, vùng nhớ cho biến
5. Các kiểu cơ bản của biến
6. Định nghĩa kiểu với typedef
7. Định nghĩa biến và gán trị cho biến
8. Hằng

1. Danh hiệu (1)

- ***Danh hiệu:*** được dùng để xác định các đại lượng khác nhau trong một chương trình như biến, hằng, hàm,... Là dãy kí tự liên nhau, gồm:
 - kí tự chữ
 - kí tự số
 - kí tự ‘_’ (*underscore character*).
- ***Qui tắc (đặt tên):***
 - Chỉ có thể bắt đầu với một kí tự *chữ* hoặc kí tự ‘_’
 - Không trùng “từ khóa”.
 - Phân biệt chữ in, chữ thường.

1. Danh hiệu (2)

Xét các ví dụ sau:

DiemMon1

Dong\$

1HK

_diemTB

123\$

int

diem HK

2. Từ khóa (key words)

- Là những “tên” đã được định nghĩa bởi ngôn ngữ, dùng cho những mục đích khác nhau:

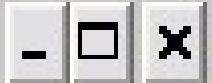
<code>void</code>	<code>char</code>	<code>if</code>	<code>return</code>	<code>static</code>
<code>do</code>	<code>int</code>	<code>else</code>	<code>sizeof</code>	<code>register</code>
<code>for</code>	<code>long</code>	<code>switch</code>	<code>enum</code>	<code>goto</code>
<code>while</code>	<code>float</code>	<code>case</code>	<code>typedef</code>	<code>struct</code>
<code>break</code>	<code>double</code>	<code>default</code>	<code>unsigned</code>	<code>continue</code>

3. Kiểu dữ liệu

- Xét tập N, Z, Q, R, C ?!
 - Kiểu dữ liệu (KDL) được xác định bởi:
 - tập giá trị, và
 - tập các phép toán tác động lên các phần tử thuộc tập giá trị ấy.
 - Đơn vị lưu trữ là *byte*. Mỗi giá trị thuộc một KDL được biểu diễn bởi một số byte nhất định.
- => Các giá trị biểu diễn được là hữu hạn.



Borland/Turbo C++ Help



File Edit Bookmark Help

Contents

Search

Back

History

≤≤

≥≥

Up

Group

Overview

Data Types

Type	Length	Range		
unsigned char	8 bits	0	to	255
char	8 bits	-128	to	127
enum	16 bits	-32,768	to	32,767
unsigned int	16 bits	0	to	65,535
short int	16 bits	-32,768	to	32,767
int	16 bits	-32,768	to	32,767
unsigned long	32 bits	0	to	4,294,967,295
long	32 bits	-2,147,483,648	to	2,147,483,647
float	32 bits	$3.4 * (10^{**}-38)$	to	$3.4 * (10^{**}+38)$
double	64 bits	$1.7 * (10^{**}-308)$	to	$1.7 * (10^{**}+308)$
long double	80 bits	$3.4 * (10^{**}-4932)$	to	$1.1 * (10^{**}+4932)$

4. Khái niệm biến, vùng nhớ cho biến (1)

- Là nơi lưu trữ dữ liệu trong bộ nhớ máy tính, được đặt bởi một tên.

int a;



- Mỗi biến chỉ có thể lưu một loại giá trị nhất định, tùy thuộc *kiểu biến* (KDL).

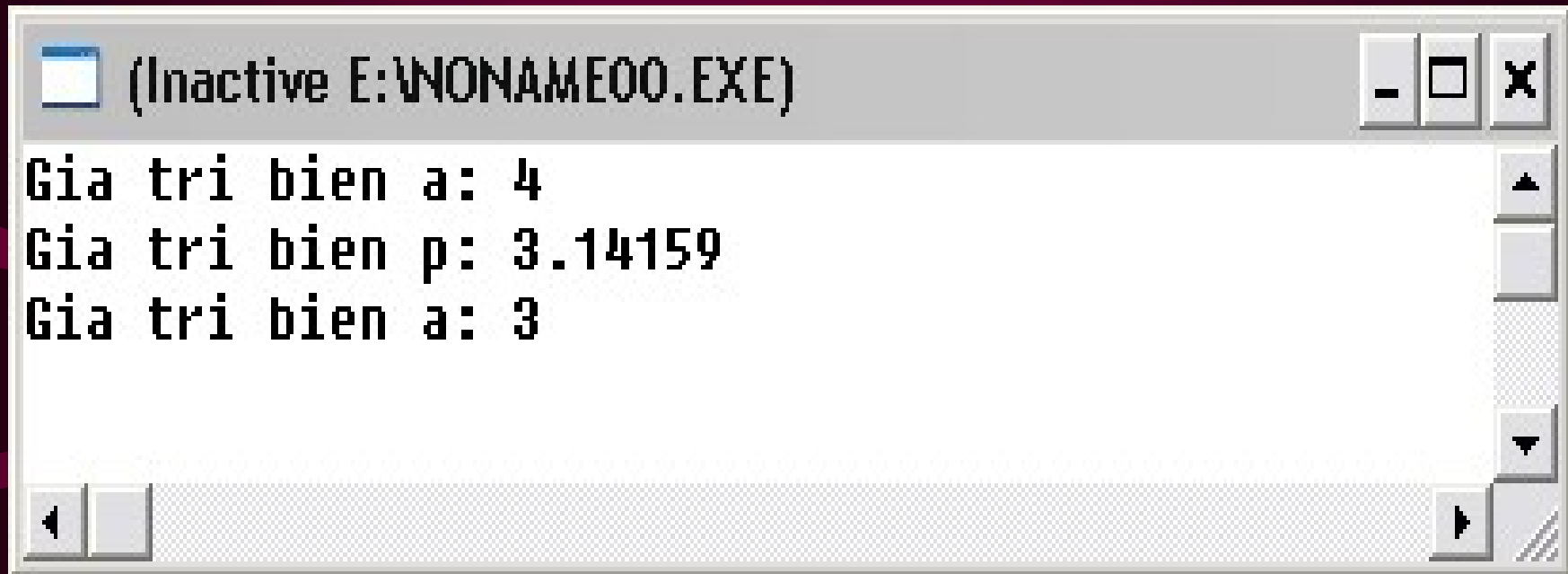
4. Khái niệm biến, vùng nhớ cho biến (2)

noname00.cpp

```
#include<iostream.h>
void main()
{
    int a= 4;
    cout<<"Gia tri bien a: "<<a<<"\n";
    double p = 3.14159;
    cout<<"Gia tri bien p: "<<p<<"\n";
    a= p;
    cout<<"Gia tri bien a: "<<a<<"\n";

}
```

4. Khái niệm biến, vùng nhớ cho biến (3)



- Giá trị của biến có thể *thay đổi*, nhưng tại mỗi thời điểm một biến chỉ lưu một giá trị.

5. Các kiểu dữ liệu cơ bản

- Kiểu số nguyên (int)
- Kiểu số thực
 - Số dấu phẩy động độ chính xác đơn (float)
 - Số dấu phẩy động độ chính xác kép (double)
- Kiểu ký tự (char)

Kiểu số nguyên (1)

char

unsigned char

int

unsigned int

long

unsigned long

Biểu diễn hằng giá trị:

1234

(*kiểu* int)

1234U

(*kiểu* unsigned int)

1234L

(*kiểu* long)

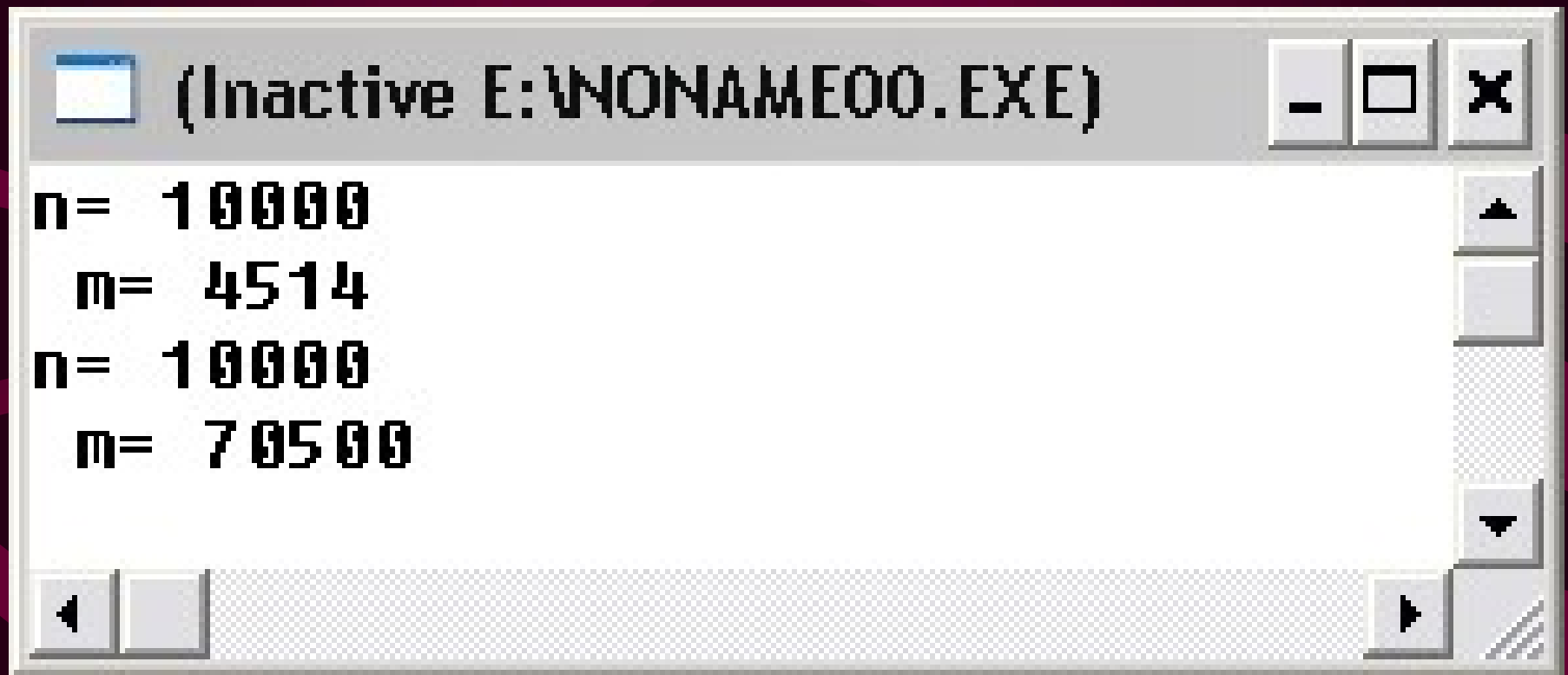
1234UL

(*kiểu* unsigned long)

 noname00.cpp

```
//Chuong trinh vi du kieu so nguyen
#include<iostream.h>
void main()
{
    int n= 10000;
    long m;
    m= 7*n + 50;
    cout<<"n= "<<n<<"\n m= "<<m<<"\n";
    m= 7L*n + 500;
    cout<<"n= "<<n<<"\n m= "<<m;
}
```

Kết quả



Kiểu số nguyên (2)

Các phép toán trên số nguyên:

$+$ $-$ $*$ $/$ $\%$

$$9/4 \rightarrow 2$$

$$1/2 \rightarrow 0$$

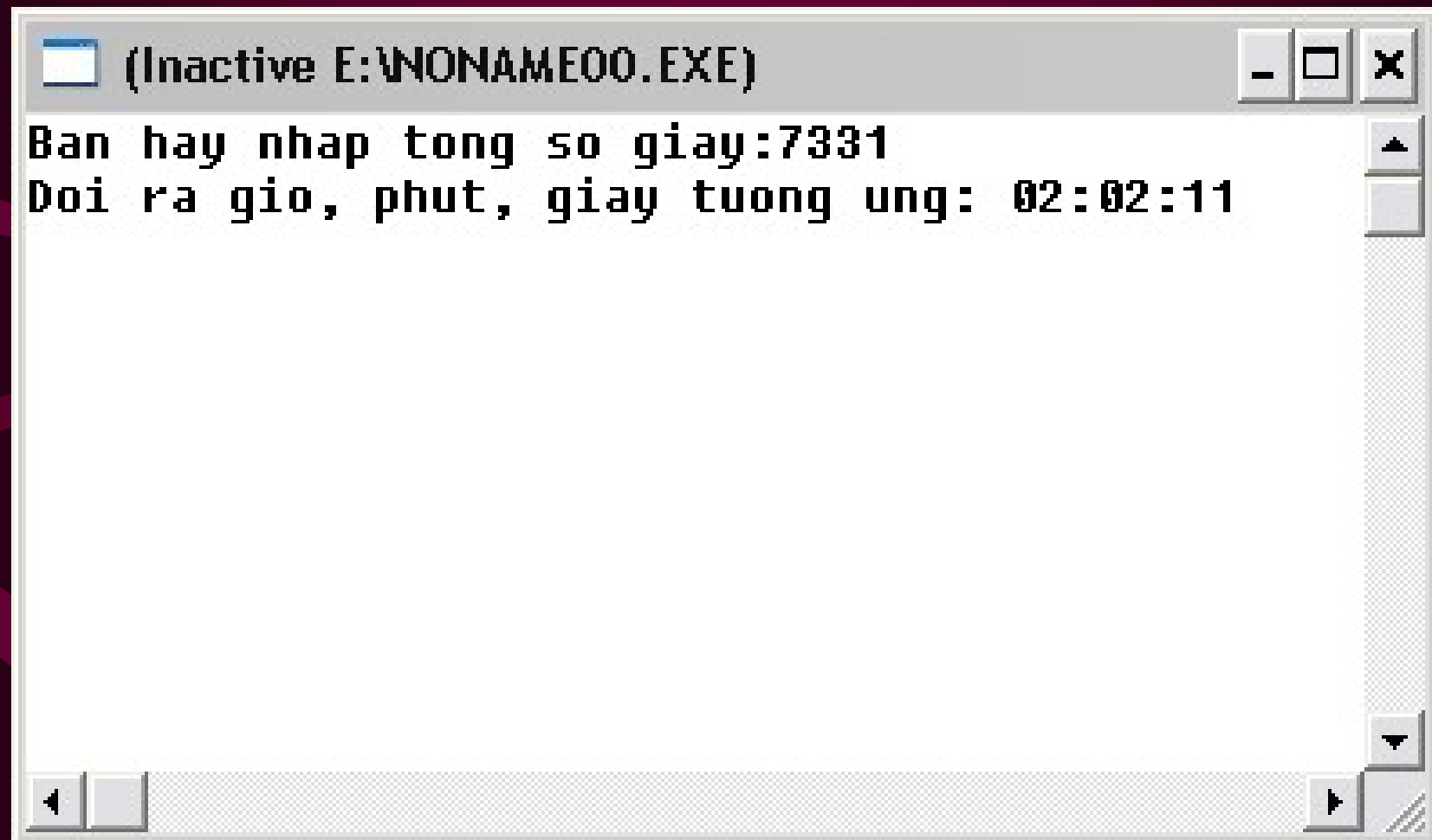
$$9\%5 \rightarrow 4$$

noname00.cpp

```
/*Ap dụng phép toan tren so nguyen
   viet chuong trinh doi gio */
#include<iostream.h>
#include<iomanip.h>

void main()
{
    unsigned long ss;
    int h, m, s;
    cout<<"Ban hay nhap tong so giay:";
    cin>>ss;
    h= ss/3600;
    m= ss%3600/60;
    s= ss%60;
    cout<<"Doi ra gio, phut, giay tuong ung: "
    cout<<setw(2)<<setfill('0')<<h<<':'
        <<setw(2)<<setfill('0')<<m<<':'
        <<setw(2)<<setfill('0')<<s<<'\n';
}
```


Kết quả



Kiểu số nguyên (3)

Các tiếp đầu ngữ: *long, short, signed, unsigned* với kiểu nguyên:

`short int` → `short`

`signed int` ≡ `int`

`unsigned int` → `unsigned`

`long int` → `long`

Kiểu số thực (1)

`float`

`double`

Hai cách biểu diễn số thực:

- *Dạng thập phân*: phần nguyên & phần phân.

`12.345`

`-0.02468`

- *Dạng chấm động*: phần định trị & phần mũ.

`1.2345e+01`

`-2.468e-02`

Biểu diễn hằng giá trị:

`12.34` (*kiểu* `double`)

`1.234e+01`

`12.34F` (*kiểu* `float`)

`1.234e+01F`

Kiểu số thực (2)

- Các phép toán trên số thực:

+ - * /

- Độ chính xác:

float: 7 chữ số thập phân

double: 15 chữ số thập phân

⇒ Kiểu double được lưu ý sử dụng:

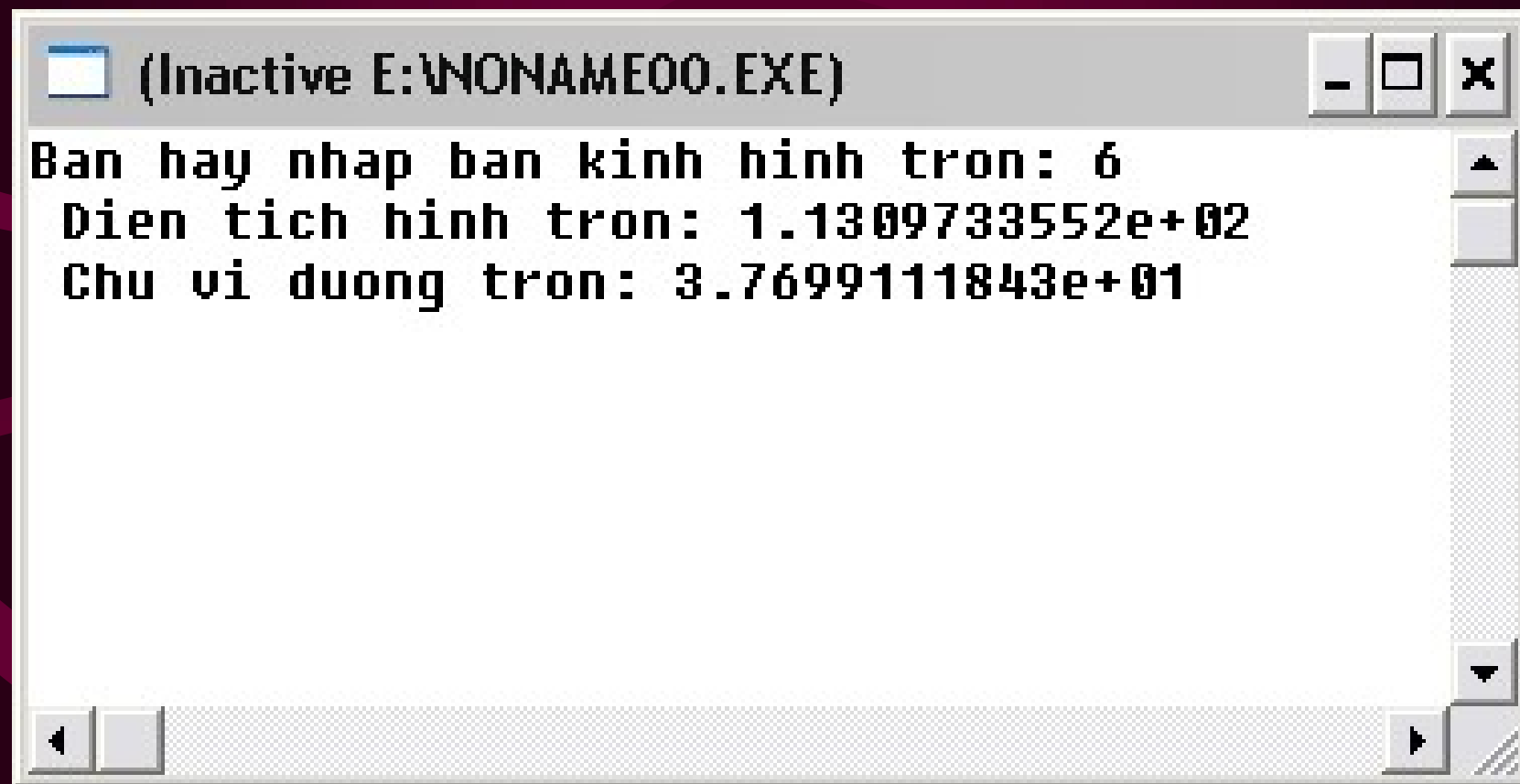
- Tính toán với số lớn.
- Cần độ chính xác cao.

```

// Tính diện tích và chu vi hình tròn
#include<iostream.h>
#include<math.h>
void main()
{
    const double PI = 4*atan(1);
    double R, S, C;
    cout<<"Ban hay nhap ban kinh hinh tron: ";
    cin>>R;
    if (R<=0)
        R= 0;
    S= PI*R*R;
    C= 2*PI*R;
    cout.precision(10); //chu y
    cout.flags(ios::scientific); //chu y
    cout<<" Dien tich hinh tron: "<<S;
    cout<<"\n Chu vi duong tron: "<<C;
}

```

Kết quả



Kiểu kí tự (1)

Biểu diễn hằng kí tự: `'a'` , `'4'` , `'@'` , ...

Tập giá trị (*1 byte, mã hoá được 256 kí tự*):

Kí tự chữ (`'a'` , `'S'` , ...)

Kí tự số (`'0'` , ... , `'9'`)

Dấu (`'@'` , `'?'` , ...)

Kí tự điều khiển (`'\n'` , `'\t'` , ...)

Kí tự đặc biệt.

Kiểu kí tự (2)

Một vài kí tự điều khiển:

`\a` alert (bell)

`\b` backspace

`\n` newline

`\t` horizontal tab

`\v` vertical tab

`\\` backslash

`\?` question mark

`\'` single quote

`\"` double quote

`\r` carriage return

Kiểu kí tự (3)

- Mỗi kí tự được lưu với một số nguyên, và theo một thứ tự nhất định gọi là *bộ mã*.
- Bộ mã được dùng phổ biến là bộ mã ASCII:

`'a'` = 97

`'A'` = 65

`'0'` = 48

`'@'` = 64

...

Bảng mã ASCII

row+col	0	1	2	3	4	5	6	7
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
8	BS	TAB	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
24	CAN	EM	SUB	ESC	FS	GS	RS	US
32	(space)	!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7
56	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G
72	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W
88	X	Y	Z	[\]	^	_
96	`	a	b	c	d	e	f	g
104	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~	DEL

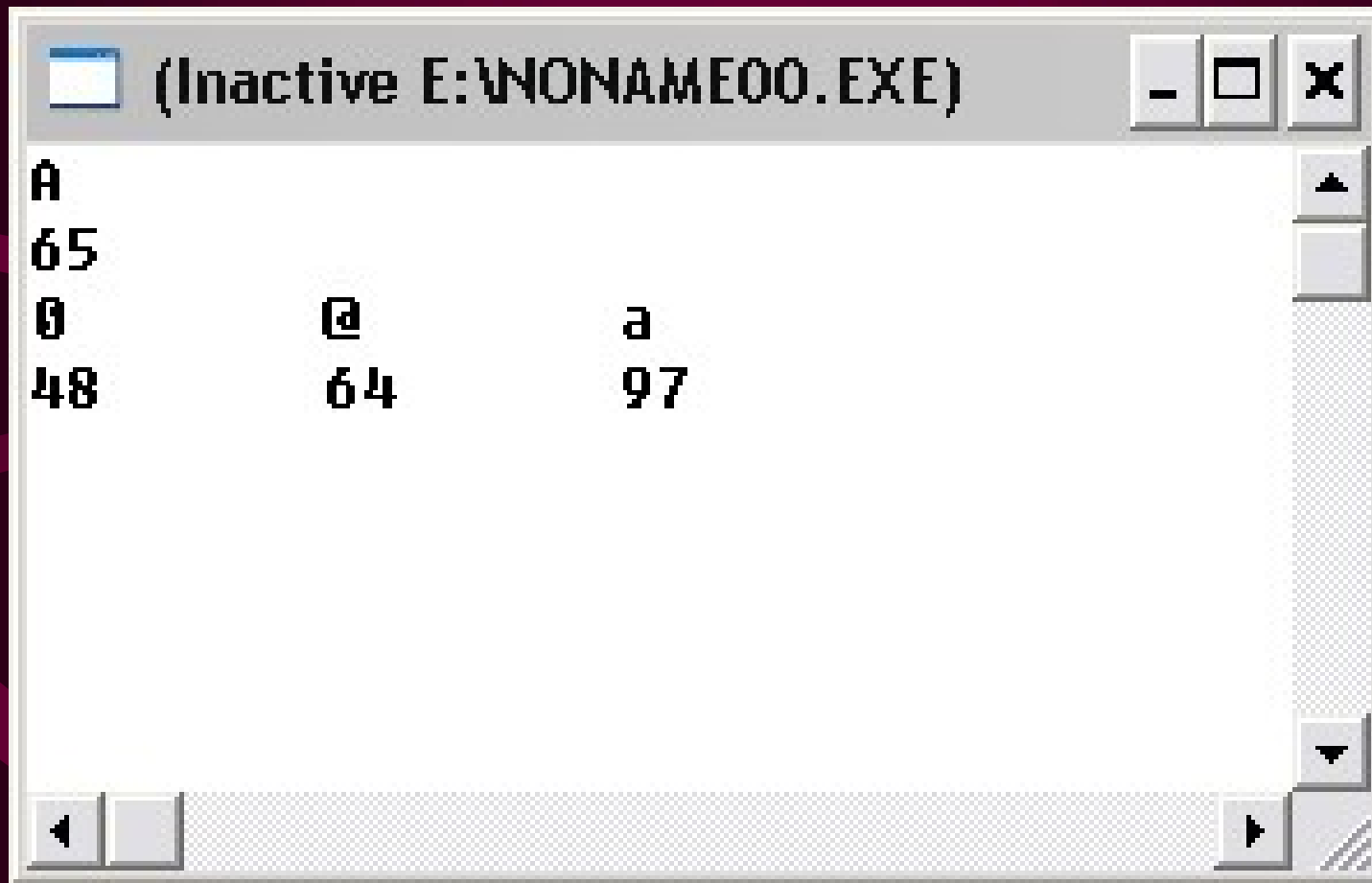
noname00.cpp

```
// Chuong trinh xem ma ASCII cua ki tu
#include<iostream.h>
void main()
{
    char t= 'A';
    cout<<t<<'\n';
    cout<<int(t)<<'\n';

    cout<<'0'<<'\t'<<'@'<<'\t'<<'a'<<'\n';

    cout<<int('0')<<'\t'<<int('@')
        <<'\t'<<int('a');
}
```

Kết quả



Kiểu kí tự (4)

Các phép toán như đối với trên số nguyên:

+ - * / %

☞ Thực hiện trên mã ASCII của kí tự tương ứng:

```
char c= 'A' ;           //c= 65
```

```
cout<<c+1;               → 66
```

```
c= c+1; cout<<c; → 'B'
```

```
c= c/2; cout<<c; → '!'
```

```
cout<<'a' - 'A' ; → 32
```

```
cout<<'8' - '3' ; → 5
```

6. Định nghĩa kiểu với typedef

- Một khai báo có thêm tiền tố *typedef* sẽ định nghĩa một tên mới cho KDL (đã có).

```
typedef KDL tenMoi;
```

- Một tên được định nghĩa theo cách này được gọi là “*định nghĩa kiểu*”.

Ví dụ

```
typedef long SoNg32;
```

```
typedef short int SoNg16;
```

```
typedef char KITU;
```

```
#include<iostream.h>
typedef unsigned long ULONG;
typedef unsigned int  UINT;
ULONG t_GiaiThua(UINT n)
{
    ULONG p= 1;
    for (UINT i= 2; i<=n; i++)
        p = p*i;
    return p;
}
void main()
{
    UINT n;
    ULONG gt;
    cout<<"Nhap so nguyen de tinh giai thua:";
    cin>>n;
    gt = t_GiaiThua(n);
    cout<<n<<"! = "<<gt;
}
```




7. Định nghĩa biến và gán trị

- Định nghĩa biến (*khai báo biến*) là đặt tên và xác định kiểu biến.
- Mọi biến cần phải được khai báo trong chương trình trước khi sử dụng.
- Để định nghĩa một biến, dạng khai báo:

KDL tenBien;

- Định nghĩa nhiều biến cùng kiểu:

KDL bien1, bien2, bienN;

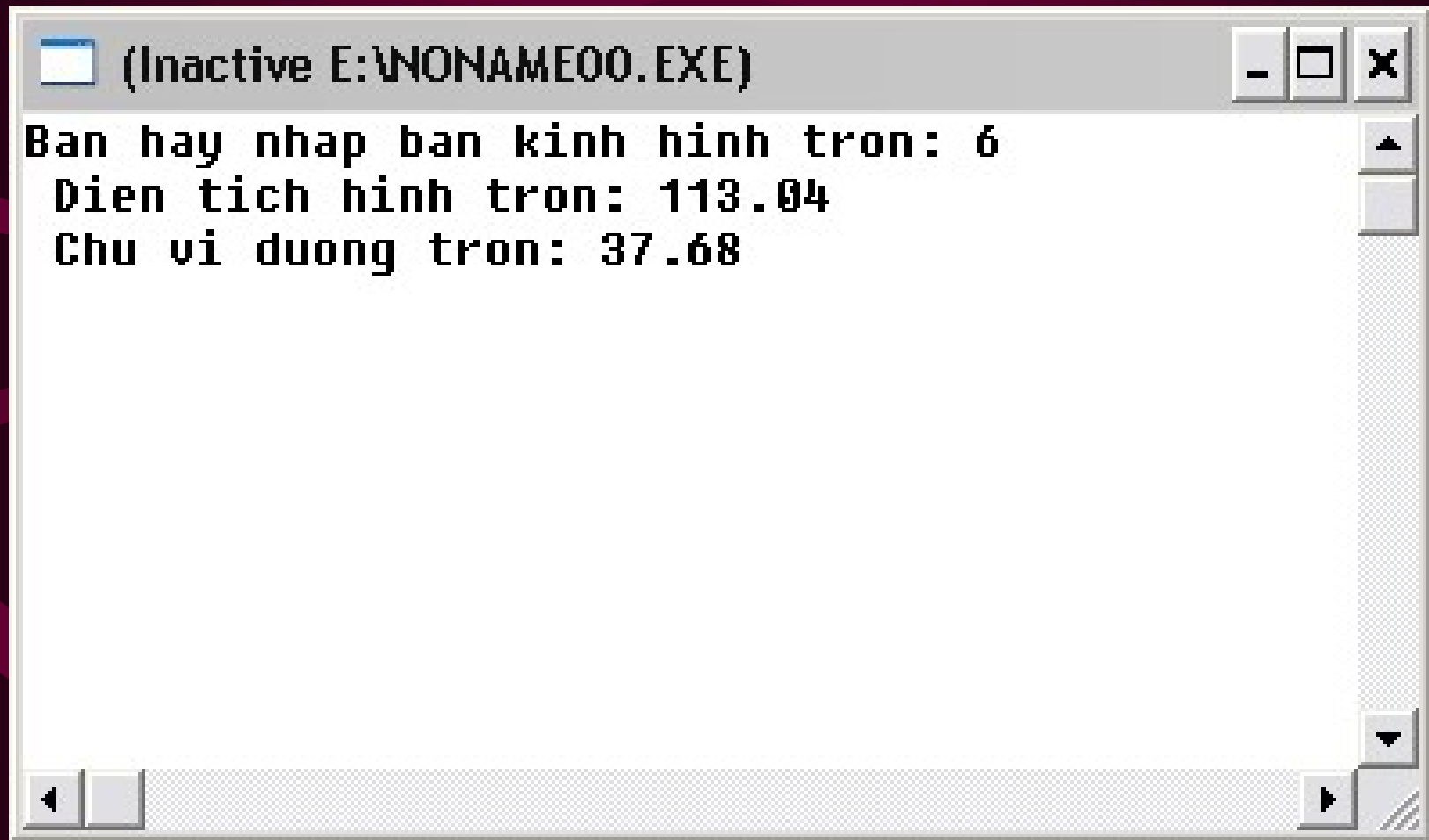
- Phép gán “=” để thay đổi giá trị biến.

tenBien = giatri;

noname00.cpp

```
/* Chương trình tính diện tích và  
chu vi hình tròn */  
#include<iostream.h>  
#include<math.h>  
void main()  
{  
    const double PI = 4*atan(1);    //khai báo hằng  
    double R, S, C;                //khai báo biến  
    cout<<"Ban hay nhap ban kinh hinh tron: ";  
    cin>>R;  
    if (R<=0)  
        R= 0;  
    S= PI*R*R;  
    C= 2*PI*R;  
    cout<<" Dien tich hinh tron: "<<S;  
    cout<<"\n Chu vi duong tron: "<<C;  
}
```

Kết quả



Gán trị cho biến (1)

- Gán liên tiếp là gán cho nhiều biến cùng lúc sau khi đã khai báo các biến. Ví dụ:

```
int a, b;
```

```
a = b = 6;
```

```
b = (a = 3) + 2;
```

- Khởi gán là gán trị cho biến ngay khi khai báo biến đó. Ví dụ:

```
double x = 1.1234;
```

Gán trị cho biến (2)

- Gán kép và khởi gán: **int a = b = 6;**
- Chú ý phân biệt:

double x= 1.0, y= 2.0, z= 1.5;

int a, b;

a = b = 6;

int a = b = 6;

int m = 3, n = 3;

8. Hằng

- Là đại lượng không đổi trong suốt quá trình thực thi của chương trình
- Phân biệt:
 - Hằng biến
 - Hằng thực sự
 - Hằng ký hiệu
- Định nghĩa hằng dùng từ khóa:
 - **const, define, enum**

Định nghĩa hằng dùng từ khóa `const`

```
const KDL TenHang = giaTriHang;
```

Ví dụ:

```
const float PI= 3.1459;
```

```
const int DVHT_m1 = 10;
```



```
// Tính diện tích và chu vi hình tròn
#include<iostream.h>
#include<math.h>
void main()
{
    const double PI = 4*atan(1);
    double R, S, C;
    cout<<"Ban hay nhap ban kinh hinh tron: ";
    cin>>R;
    if (R<=0)
        R= 0;
    S= PI*R*R;
    C= 2*PI*R;
    cout.precision(10); //chu y
    cout.flags(ios::scientific); //chu y
    cout<<" Dien tich hinh tron: "<<S;
    cout<<"\n Chu vi duong tron: "<<C;
}
```

d:\borland\work\vd_const.cpp

```
#include<iostream.h>
void main()
{
    const int dvht_m1= 8, dvht_m2= 10;
    float diem1, diem2, dtb;
    cout<<"Ban hay nhap diem tong ket mon 1 & 2:";
    cin>>diem1>>diem2;
    dtb = (diem1*dvht_m1 + diem2*dvht_m2)/(dvht_m1 + dvht_m2);
    cout.precision(2);
    cout<<"Diem trung binh tinh duoc:" << dtb;
}
```

Khi không định kiểu hằng, hằng có kiểu mặc định là kiểu int

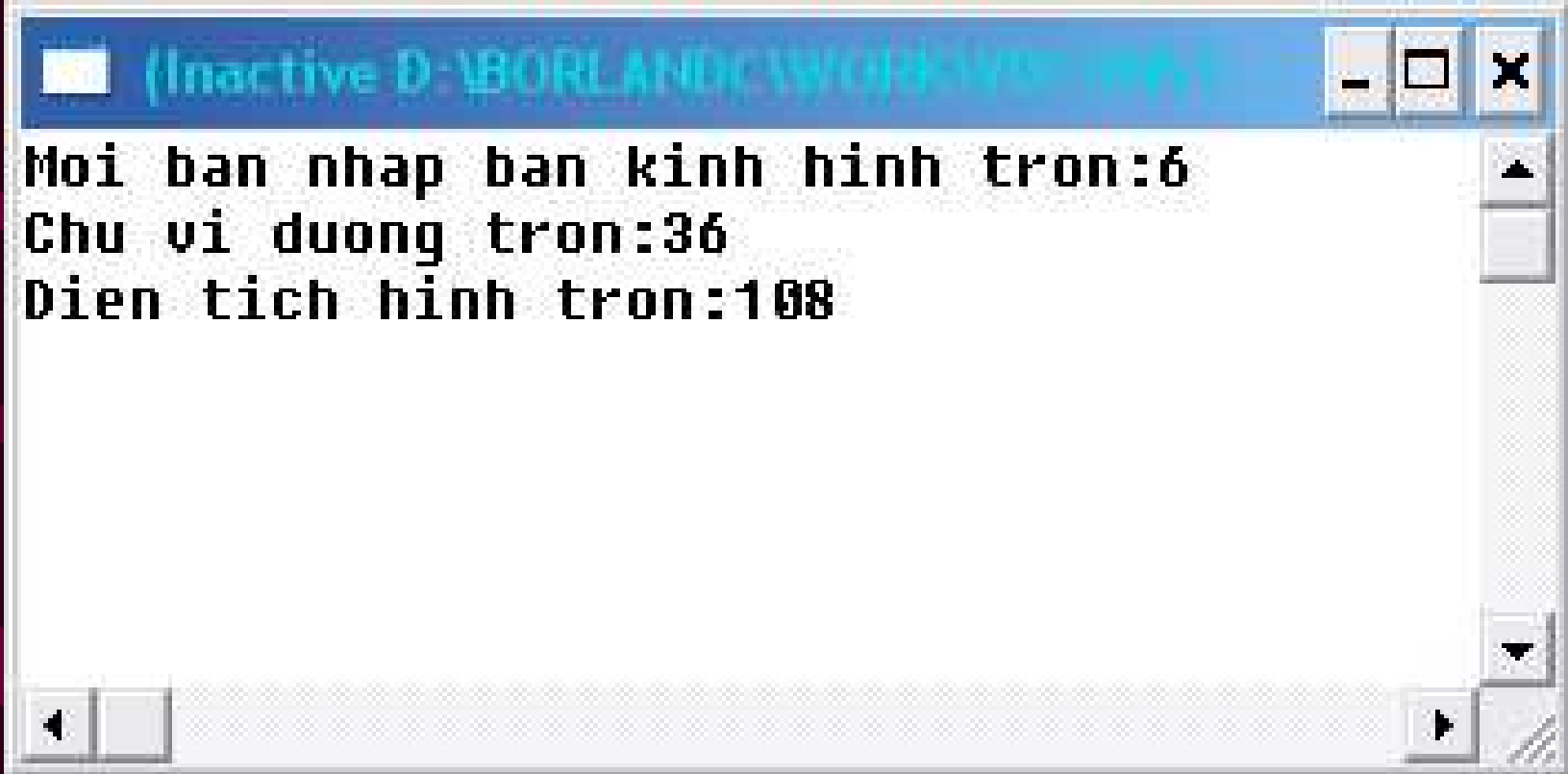
d:\borlandc\work\vd_const.cpp

```
#include<iostream.h>
void main()
{
    const dvht_m1= 8, dvht_m2= 10;
        //Khi không định kiểu hằng, mặc định kiểu là int
    float diem1, diem2, dtb;
    cout<<"Ban hay nhap diem tong ket mon 1 & 2:";
    cin>>diem1>>diem2;
    dtb = (diem1*dvht_m1 + diem2*dvht_m2)/(dvht_m1 + dvht_m2);
    cout.precision(2);
    cout<<"Diem trung binh tinh duoc:" << dtb; |
}
```

d:\borland\work\vdconst2.cpp

```
#include<iostream.h>
void main()
{
    const PI = 3.14159;
        //"Quên" dinh kieu hang => mac dinh, kieu int
    double R, S, C;
    cout<<"Moi ban nhap ban kinh hinh tron:";
    cin>>R;
    if (R<0)
        R= 0;
    C= 2*PI*R;      S= PI*R*R;
    cout<<"Chu vi duong tron:"<<C<<"\n";
    cout<<"Dien tich hinh tron:"<<S<<"\n";
}
```

Kết quả



The screenshot shows a window titled "(Inactive) D:\BORLAND\WORK\VC\APP1" with standard Windows window controls (minimize, maximize, close). The window contains a text area with the following output:

```
Moi ban nhap ban kinh hinh tron:6  
Chu vi duong tron:36  
Dien tich hinh tron:108
```

The window also features a vertical scrollbar on the right and horizontal/vertical scrollbars at the bottom.

Định nghĩa hằng ký hiệu, dùng từ khóa define

```
#define TenHang giaTriHang
```

Chú ý: *Không dùng ‘;’*
 Không dùng phép gán =
 Một định nghĩa chỉ một hằng

Ví dụ:

```
#define PI    3.1459
```

```
#define DVHT_m1    10
```


```
#define DVHT_m2    8
```

 d:\borlandc\work\vdconst2.cpp

```
#include<iostream.h>

#define PI 3.14159
void main()
{
    double R, S, C;
    cout<<"Moi ban nhap ban kinh hinh tron:";
    cin>>R;
    if (R<0)
        R= 0;
    C= 2*PI*R;          S= PI*R*R;
    cout<<"Chu vi duong tron:"<<C<<"\n";
    cout<<"Dien tich hinh tron:"<<S<<"\n";
}
```

Kết quả



The screenshot shows a window titled "(Inactive D:\BORLAND\WORK\VC\BIN\17.337)". The window contains the following text:

```
Moi ban nhap ban kinh hinh tron:6  
Chu vi duong tron:37.69908  
Dien tích hình tron:113.09724
```

The window has standard Windows-style controls: minimize, maximize, and close buttons in the top right corner, and scrollbars on the right and bottom.

Hằng liệt kê, dùng từ khóa enum (1)

```
enum { hang1, hang2, ..., hangN }
```

- Dùng khi có muốn định nghĩa nhiều hằng nguyên.
- Mặc định các giá trị hằng liên tiếp nhau, bắt đầu là 0.

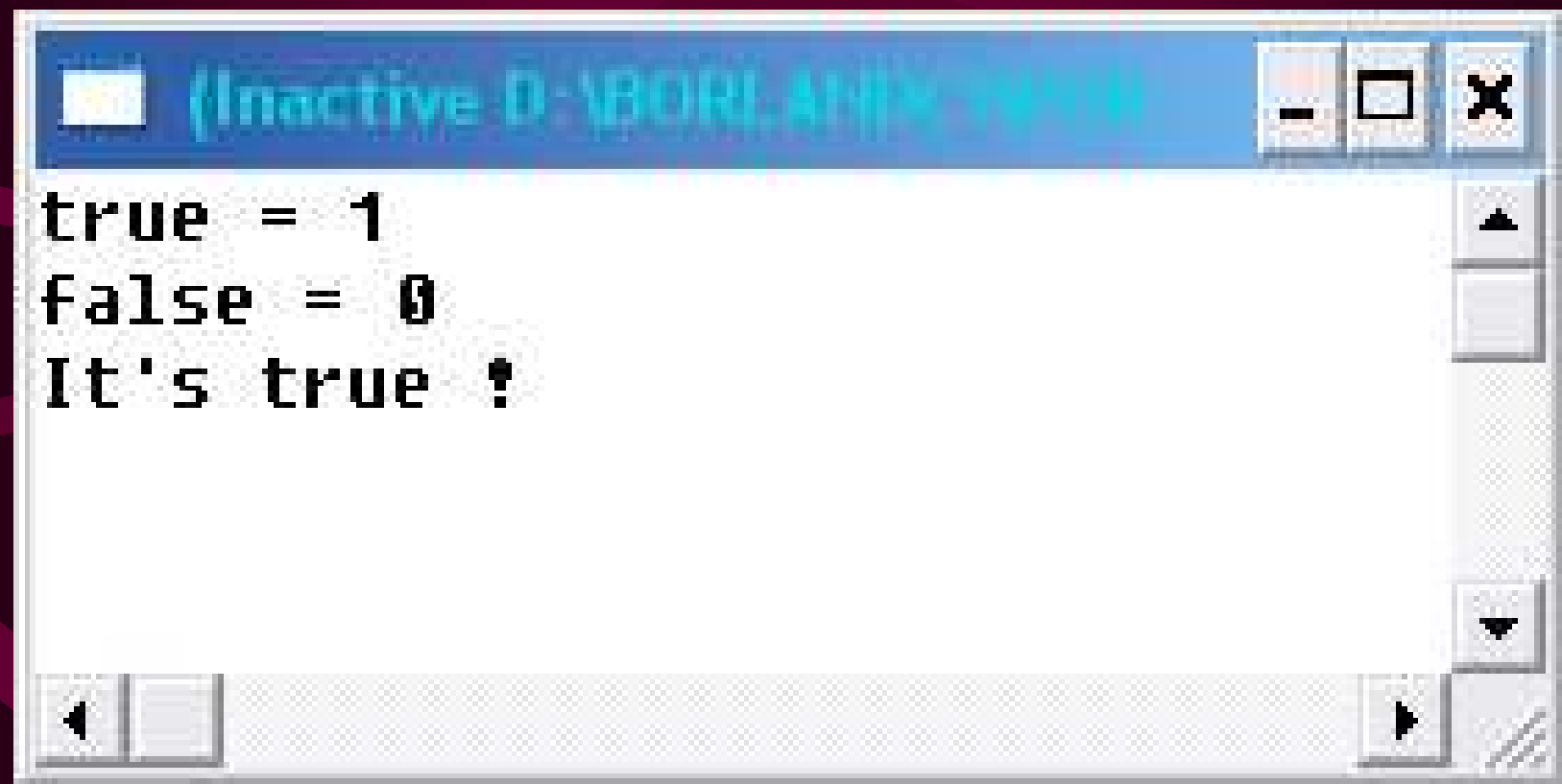
```
enum { false, true };
```

```
enum { auto, remote, hand };
```

 d:\borland\work\vd_enum.cpp

```
#include<iostream.h>
enum {false, true};
void main()
{
    cout<<"true = "<<true<<"\n";
    cout<<"false = "<<false<<"\n";
    int a= 1;
    if (a == true)
        cout<<"It's true !";
}
```

Kết quả



The image shows a screenshot of a Borland Turbo Pascal IDE window. The window title bar reads "(Inactive) D:\BORLAND\TWIN". The main text area displays the following output:

```
true = 1  
false = 0  
It's true !
```

The window includes standard Windows-style controls: minimize, maximize, and close buttons in the top right corner, and scrollbars on the right and bottom edges.

Hằng liệt kê, dùng từ khóa enum (2)

- Định trị bắt đầu của danh sách hằng:

```
enum { auto= -1, remote, hand };
```

```
enum { Mon= 2, Tue, Wed, Thu,  
      Fri, Sat, Sun };
```

 noname00.cpp

```
#include<iostream.h>
enum { Mon=2, Tue, Wed, Thu, Fri, Sat, Sun};
void main()
{
    cout<< Mon << endl
        << Tue << endl
        << Wed << endl
        << Thu << endl
        << Fri << endl
        << Sat << endl
        << Sun << endl;
}
```

Kết quả



Hằng liệt kê, dùng từ khóa enum (3)

- Định trị cho từng tên hằng:


```
enum { auto=-1, remote= 2, hand= 5 };
```

```
enum { start= 'A', mid='M', end= 'Z' };
```

d:\borlandc\work\vd_enum3.cpp

```
#include<iostream.h>
enum {start = '0', mid='5', end='9'};
enum {dvht1= 8, dvht2= 10, dvht3= 5};
void main()
{
    cout<<"Gia tri cua hang start la: "<<start<<"\n";
    cout<<"Gia tri cua hang midle la: "<<mid<<"\n";
    cout<<"Gia tri cua hang end la: "<<end<<"\n";
    cout<<"Gia tri cua hang dvht2 la: "<<dvht2<<"\n";
}
```


Kết quả



The image shows a screenshot of a console window from a Borland C++ IDE. The window title is "(Inactive) D:\BORLANDC\WORKING_1\BIN\1421". The console displays four lines of text in a monospaced font, representing the output of a program. The text is as follows:

```
Gia tri cua hang start la: 48  
Gia tri cua hang midle la: 53  
Gia tri cua hang end la: 57  
Gia tri cua hang dvht2 la: 10
```

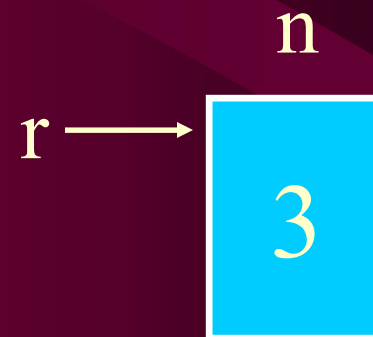
Tham chiếu (1)

- Là một *tên gọi khác* để truy cập đến cùng địa chỉ (vùng nhớ) với biến đã có.
- Mẫu khai báo:

KDL & ref = var;

Ví dụ:

```
int n = 3;  
int &r = n;
```



Tham chiếu (2)

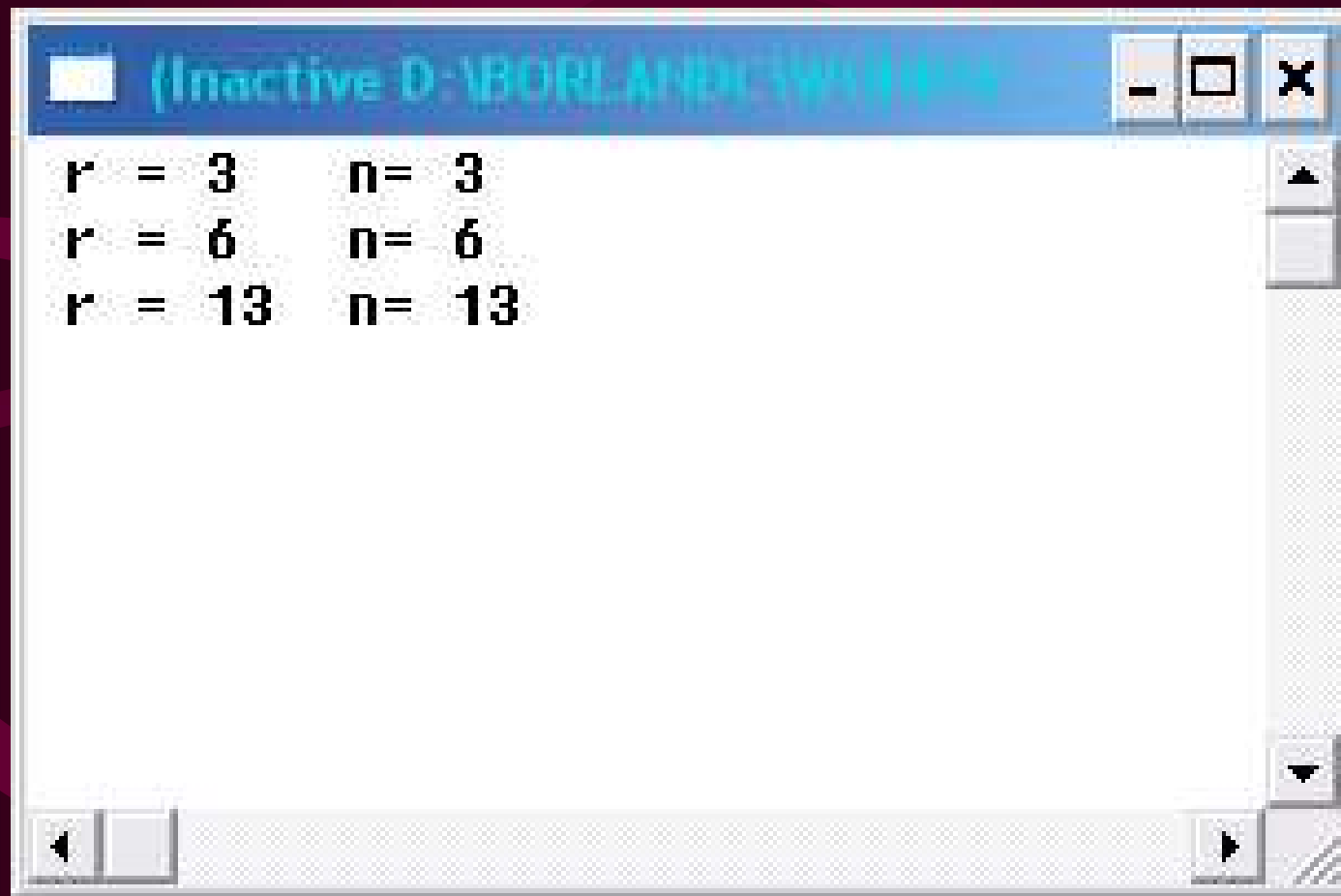
Được sử dụng chính:

- Đối với tham số của hàm.
- Trong kiểu trả về của hàm.
- Cho các phép toán “*nạp chồng*”.

 d:\borlandc\work\vd_ref.cpp

```
#include<iostream.h>
void main()
{
    int n = 3;
    int &r = n;
    cout<<" r = "<<r<<"\t n= "<<n;
    r= r*2;
    cout<<"\n r = "<<r<<"\t n= "<<n;
    n= n + 7;
    cout<<"\n r = "<<r<<"\t n= "<<n;
}
```

Kết quả

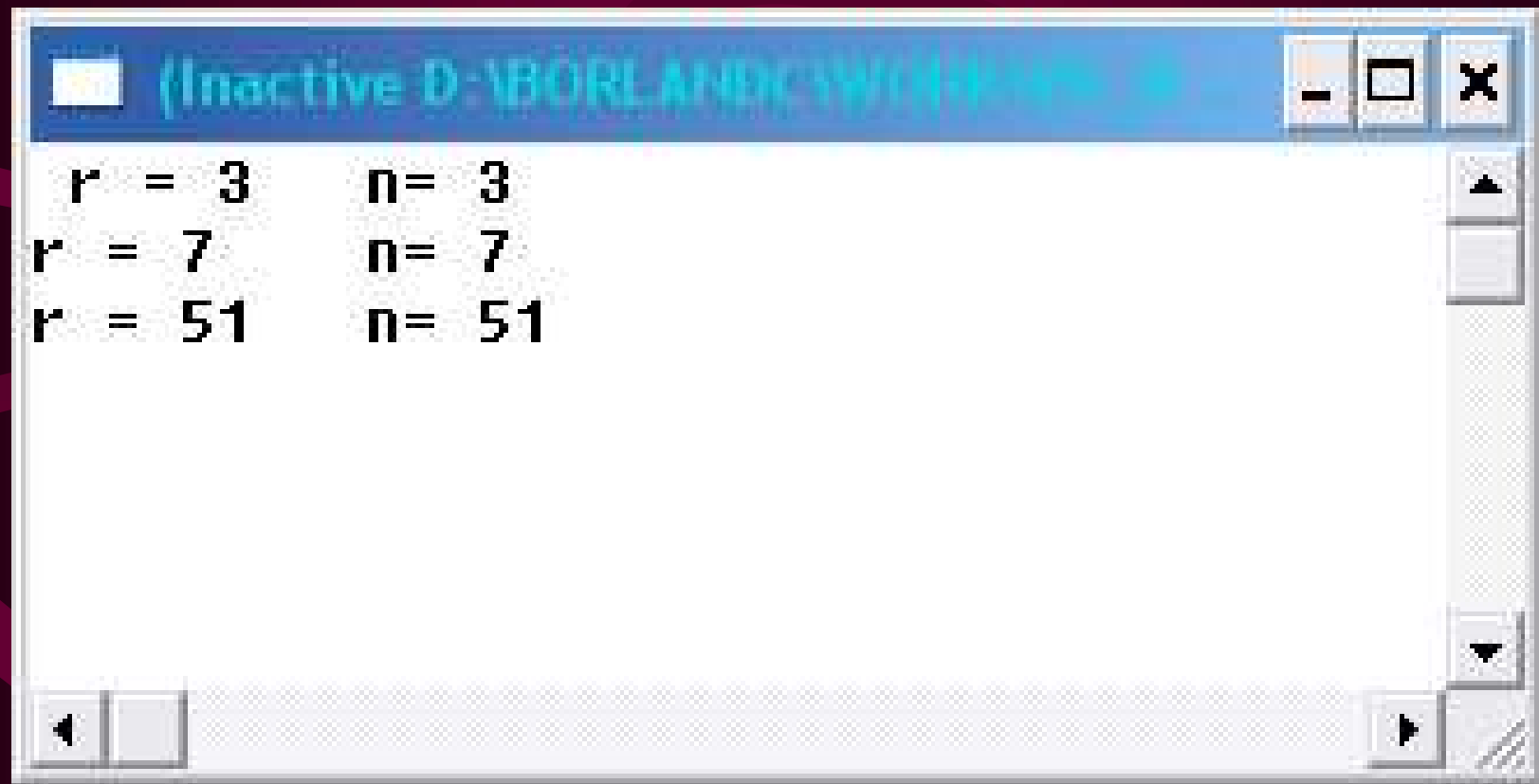




d:\borlandc\work\vd_ref2.cpp

```
#include<iostream.h>
void main()
{
    int n = 3, m= 7;
    int &r = n;
    cout<<" r = "<<r<<"\t n= "<<n;
    r= m;
    cout<<"\nr = "<<r<<"\t n= "<<n;
    n= m*m+2;
    cout<<"\nr = "<<r<<"\t n= "<<n;
}
```

Kết quả

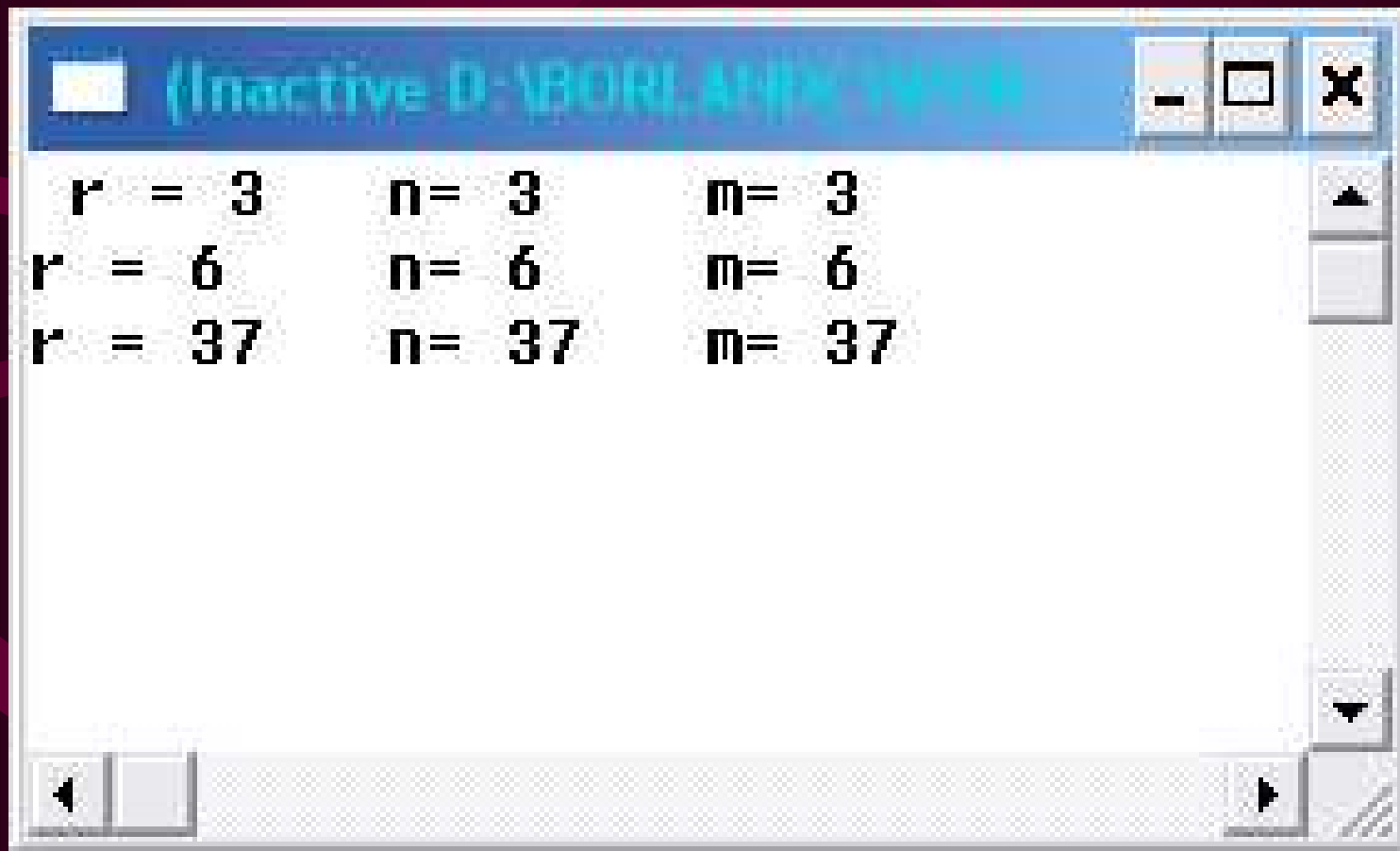


 d:\borlandc\work\vd_ref3.cpp

```
#include<iostream.h>
void main()
{
    int n = 3;
    int &r = n;        //tham chieu
    int &m= r;         //cung tham chieu
    cout<<" r = "<<r<<"\t n= "<<n<<"\t m= "<<m;
    r= r*2;
    cout<<"\nr = "<<r<<"\t n= "<<n<<"\t m= "<<m;
    n= n*n+1;
    cout<<"\nr = "<<r<<"\t n= "<<n<<"\t m= "<<m;

}
```


Kết quả



$r = 3$	$n = 3$	$m = 3$
$r = 6$	$n = 6$	$m = 6$
$r = 37$	$n = 37$	$m = 37$

Tham chiếu (2)

- Không tham chiếu đến biến *khác kiểu*.

```
double x;  
int &n = x;      ///  
                ///
```

- Không tham chiếu đến hằng.

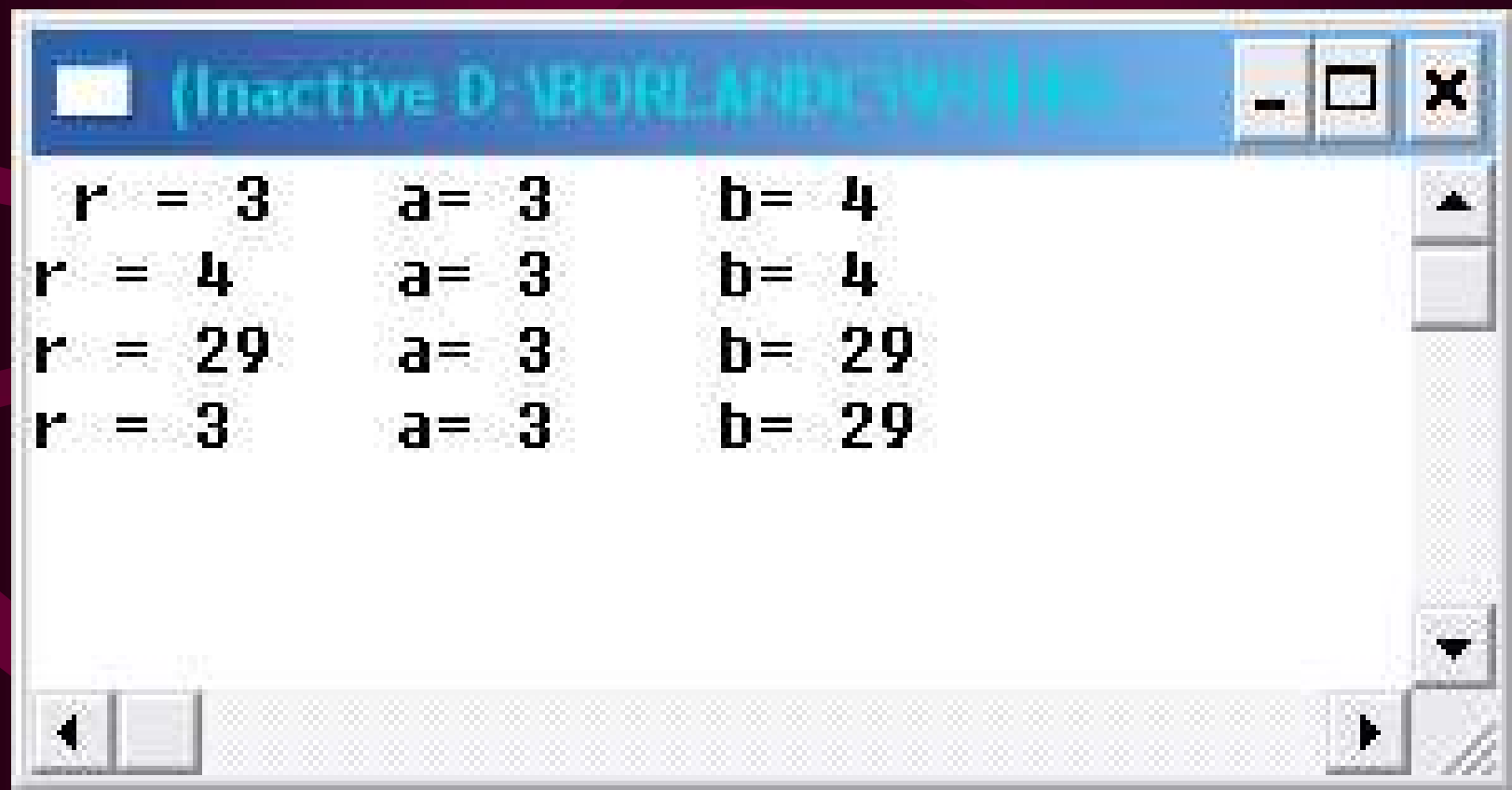
```
const int a = 5;  
int &r = a;      ///  
                ///  
int &t = 7;      ///  
                ///
```

```
#include<iostream.h>
void main()
{
    int a= 3, b= 4;
    int &r = a;        //tham chieu a
    cout<<" r = "<<r<<"\t a= "<<a<<"\t b= "<<b;

    &r= &b;            //tham chieu b
    cout<<"\nr = "<<r<<"\t a= "<<a<<"\t b= "<<b;
    b= 7*b+1;
    cout<<"\nr = "<<r<<"\t a= "<<a<<"\t b= "<<b;

    &r= &a;            //tham chieu a
    cout<<"\nr = "<<r<<"\t a= "<<a<<"\t b= "<<b;
}
```

Kết quả



The screenshot shows a window from a Borland C++ IDE. The title bar reads "(Inactive) D:\BORLAND\EXAMPLES...". The main area contains a table of values for variables r, a, and b across four rows. The window has standard Windows-style controls (minimize, maximize, close) and a vertical scrollbar on the right.

r = 3	a = 3	b = 4
r = 4	a = 3	b = 4
r = 29	a = 3	b = 29
r = 3	a = 3	b = 29

Bài tập 1

xx

BANKINH

2vars

dong_ \$

var-2

chieu dai

ban_kinh

chieu rong

x^2

DienTich

Bài tập 2

```
int a, b, dienTich, chuVi;  
const double Pi = 3.14159  
const long rate = 16019L;  
float chieu dai, chieu rong;  
char t= 'a';  
char ho= 'nguyen';  
int a= b= 2, S, C;
```

Bài tập 3

```
#define PI= 3.14159
```

```
double R= 2, dT, cV;
```

```
double diem_m_1, diem-mon-2, dTB;
```

```
const dvht 1= 3, dvht 2= 4;
```

```
char ten= "nam";
```

```
long tien = 100000
```