

CHƯƠNG 2

CÁC KHÁI NIỆM CƠ BẢN

Th.S Dương Thị Thùy Vân

Khoa CNTT & TỬĐ

Nội dung

1. Ngôn ngữ lập trình
2. Chương trình dịch
3. Soạn thảo mã nguồn-Biên dịch-Liên kết và thực thi
4. Ví dụ chương trình C
5. Các thành phần của chương trình C đơn giản

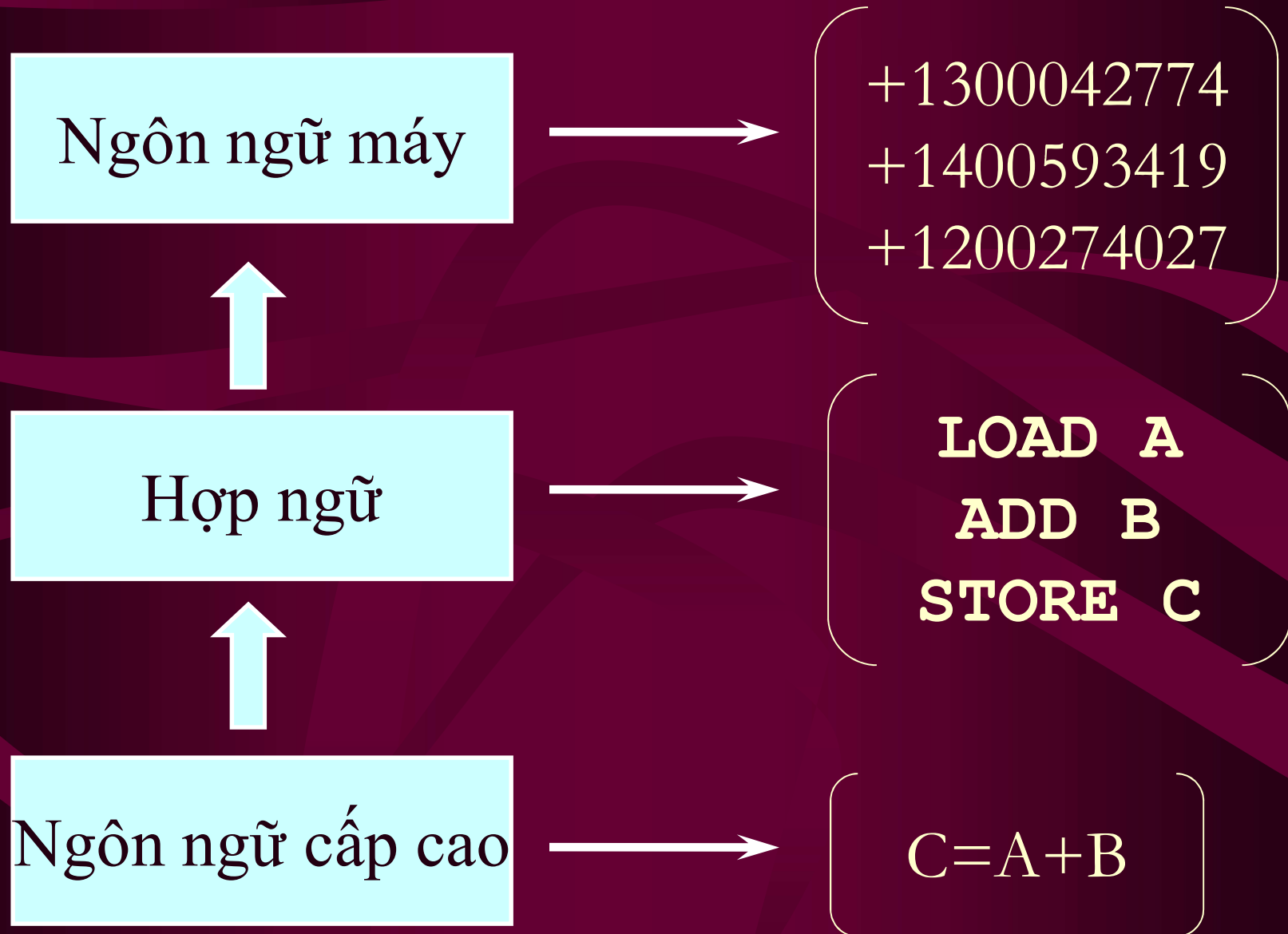
1. Ngôn ngữ lập trình (1)

- Con người liên lạc với nhau dùng:
 - ngôn ngữ tự nhiên: các mẫu từ ngữ và âm thanh
- Con người “nói chuyện” với máy tính dùng:
 - Ngôn ngữ lập trình: tập từ ngữ và ký hiệu
 - Tuân theo các luật được gọi là cú pháp (syntax)
- Có rất nhiều ngôn ngữ lập trình đang được sử dụng.

1. Ngôn ngữ lập trình (2)

- Dựa vào mức độ chi tiết hóa việc mô tả các thao tác, người ta chia ngôn ngữ lập trình thành các lớp:
 - Ngôn ngữ máy,
 - Hợp ngữ,
 - Ngôn ngữ cấp cao.
- Các ngôn ngữ cấp cao gần với ngôn ngữ tự nhiên nên rất tiện lợi cho việc mô tả các thao tác và dễ học, dễ nhớ.

1. Ngôn ngữ lập trình (3)



Ngôn ngữ máy (1)

- Là ngôn ngữ nền tảng của bộ vi xử lý, còn được gọi là mã máy.
- Tập lệnh của ngôn ngữ máy phụ thuộc vào loại vi xử lý nên ngôn ngữ máy sẽ khác nhau trên những máy tính có sử dụng bộ vi xử lý khác nhau.
- Các chương trình được viết bằng các loại ngôn ngữ khác cuối cùng đều được chuyển thành ngôn ngữ máy trước khi chương trình đó được thi hành.

Ngôn ngữ máy (2)

- Ưu điểm viết chương trình bằng ngôn ngữ máy:
 - điều khiển máy tính trực tiếp
 - đạt được chính xác điều mình muốn làm.
 - hiệu quả về tốc độ thi hành, kích thước chương trình nhỏ

=> ngôn ngữ máy cho phép khai thác triệt để khả năng của máy tính.

- Bất lợi của chương trình ngôn ngữ máy
 - Tốn rất nhiều thời gian để viết,
 - Rất khó đọc, khó theo dõi để tìm lỗi
 - Chỉ chạy được trên những máy tính có cùng bộ vi xử lý.

=> ngôn ngữ máy được gọi là ngôn ngữ cấp thấp.

Hợp ngữ

- Tương tự như ngôn ngữ máy nhưng sử dụng các ký hiệu gọi nhớ để biểu diễn cho mã lệnh của máy.
- Cho phép định địa chỉ hình thức (dùng tên hoặc ký hiệu để tham chiếu tới một vị trí bộ nhớ) thay vì phải sử dụng địa chỉ thực sự (bằng con số nhị phân) như ngôn ngữ máy.
- Được phát triển nhằm giúp lập trình viên dễ nhớ các lệnh của chương trình.
- Các chương trình hợp ngữ được chuyển sang mã máy thông qua trình hợp dịch (assembler).

Ngôn ngữ cấp cao

- Hợp ngữ vẫn còn rất gần với từng thiết kế của máy tính.
- Cần có những ngôn ngữ lập trình gần với ngôn ngữ tự nhiên hơn, được gọi là ngôn ngữ cấp cao.
- Ngôn ngữ cấp cao bao gồm: danh từ, động từ, ký hiệu toán học, liên hệ và thao tác luận lý. Các yếu tố này có thể được liên kết với nhau tạo thành câu lệnh.
- Ưu điểm viết chương trình bằng ngôn ngữ cấp cao:
 - Dễ đọc và dễ học
 - Không phụ thuộc vào máy tính

Các thành phần của ngôn ngữ lập trình

- Mỗi ngôn ngữ lập trình thường có ba thành phần cơ bản:
 - **Bảng chữ cái:** là tập hợp các kí tự được dùng khi viết chương trình, ngoài các kí tự này không được phép dùng bất kì kí tự nào khác.
 - **Cú pháp:** là bộ quy tắc để viết chương trình. Dựa vào chúng, người lập trình và chương trình dịch biết được tổ hợp nào của các kí tự trong bảng chữ cái là hợp lệ và tổ hợp nào là không hợp lệ. Nhờ đó, có thể mô tả chính xác thuật toán để máy thực hiện.
 - **Ngữ nghĩa:** xác định ý nghĩa thao tác cần phải thực hiện, ứng với mỗi tổ hợp kí tự và dựa vào ngữ cảnh của nó.

Ví dụ

- Hầu như các ngôn ngữ lập trình đều có kí tự + chỉ phép cộng. Xét các biểu thức:
 $A + B$ (1)
 $I + J$ (2)
- Giả thiết A, B là các biến thực và I, J là các biến nguyên.
- Khi đó dấu trong biểu thức (1) sẽ được hiểu là cộng hai số nguyên, dấu + trong biểu thức (2) sẽ được hiểu là cộng hai số thực.
- Như vậy, cú pháp cho biết cách viết một chương trình hợp lệ,
- Còn ngữ nghĩa xác định tính chất, thuộc tính của các tổ hợp kí tự trong chương trình.

2. Chương trình dịch

- Chuyển đổi chương trình từ NN cấp cao (hợp ngữ) thành NN máy.
- Có hai kỹ thuật chính:
 - *Trình biên dịch (compiler),*
 - *Trình thông dịch (interpreter).*

Trình biên dịch

- Chuyển đổi toàn bộ chương trình sang ngôn ngữ máy và lưu kết quả vào đĩa để có thể thi hành về sau.
 - Chương trình nguồn (source program) là chương trình ngôn ngữ cấp cao được chuyển đổi.
 - Chương trình đối tượng (object program) là chương trình ngôn ngữ máy được tạo ra.
- Thực hiện
 - Duyệt, kiểm tra cú pháp chương trình,
 - Kiểm tra logic và đảm bảo các dữ liệu sử dụng được định nghĩa hợp lý,
 - Phát hiện và tạo ra một danh sách lỗi cú pháp của các mệnh đề (statement).
- Phương pháp dịch này thuận tiện cho các chương trình ổn định và cần thực hiện nhiều lần.

Trình thông dịch

- Lần lượt dịch và thực hiện từng câu lệnh một.
- Mỗi lần chạy chương trình là mỗi lần chương trình nguồn được thông dịch sang ngôn ngữ máy.
- Ưu điểm
 - Có thể chạy một chương trình vẫn còn lỗi cú pháp.
- Nhược điểm
 - Chậm hơn các chương trình được biên dịch.
- Phương pháp dịch này thích hợp cho môi trường đối thoại giữa người và hệ thống.
- Đa số các ngôn ngữ cấp cao đều dùng trình biên dịch.

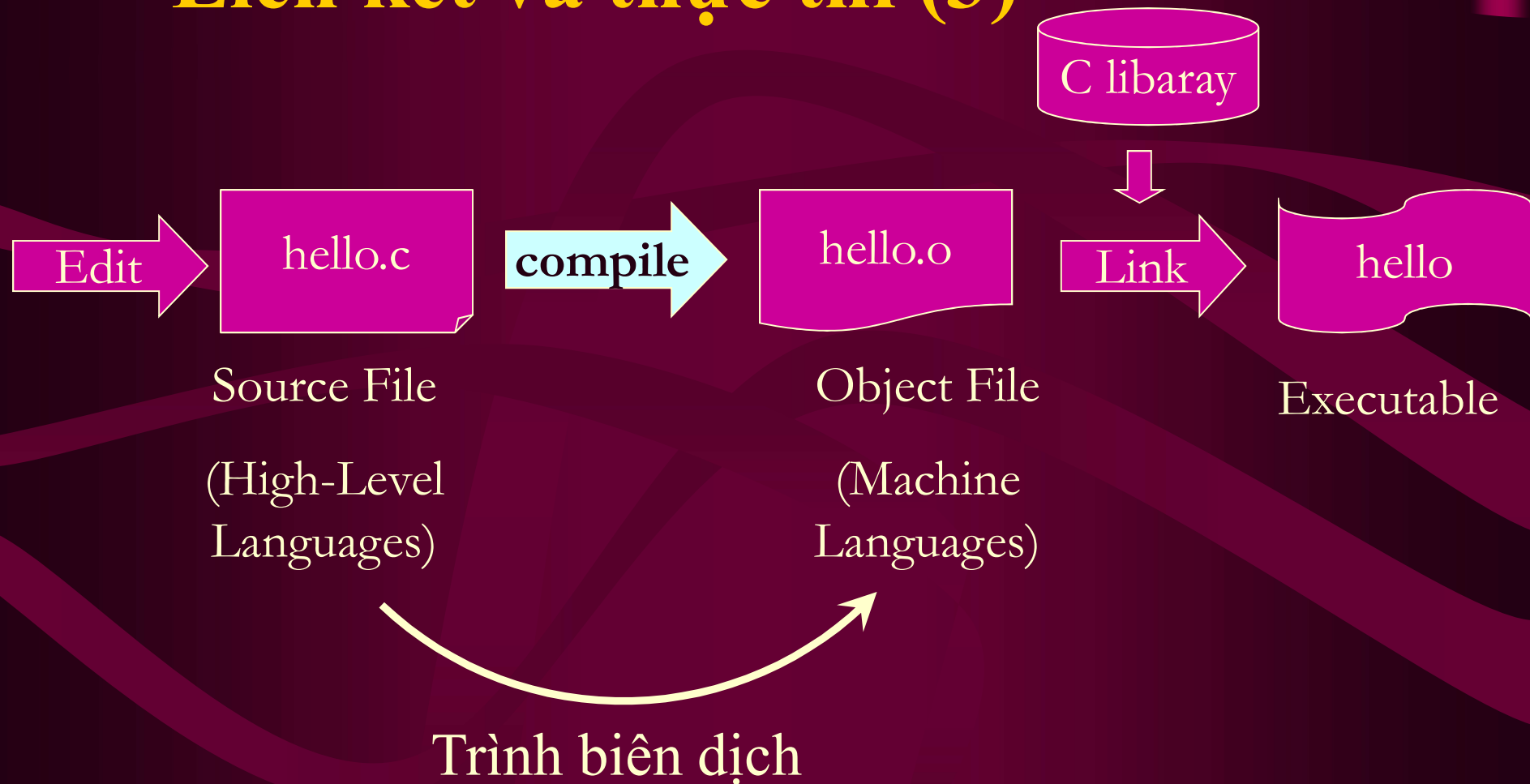
3. Soạn thảo mã nguồn – Biên dịch – Liên kết và thực thi (1)

- Mỗi ngôn ngữ lập trình có một vài môi trường lập trình tương ứng.
- Ví dụ ngôn ngữ C có các môi trường lập trình Turbo C, Borland C, Microsoft Visual C++, ...
- Môi trường lập trình cung cấp các dịch vụ như:
 - Soạn thảo mã nguồn,
 - Lưu trữ, tìm kiếm,
 - Xác định loại lỗi nếu có, chỉ rõ lỗi ở câu lệnh nào,
 - Cho xem các kết quả trung gian,
 - ...

3. Soạn thảo mã nguồn – Biên dịch – Liên kết và thực thi (2)

- Các môi trường lập trình khác biệt nhau ở các loại dịch vụ mà nó có thể cung cấp.
- Đặc biệt là các dịch vụ mở rộng, nâng cấp, tăng cường các khả năng mới cho ngôn ngữ lập trình.
- Khi biên dịch chương trình nguồn, người lập trình sẽ phát hiện được các lỗi cú pháp.
- Khi liên kết và thực thi chương trình trên dữ liệu cụ thể thì mới phát hiện được các lỗi ngữ nghĩa.

3. Soạn thảo mã nguồn – Biên dịch – Liên kết và thực thi (3)



4. Ví dụ chương trình C

```
/* Chương trình tính tổng các số tự nhiên từ 1 đến N */
#include<iostream.h>
int sum(int n);           //khai bao ham
void main()
{
    int S, n;              //biến lưu tổng và số n được cho ban đầu
    cout<<"Moi nhap vao so n: ";
    cin>>n;
    S = sum(n);           //gọi hàm tính tổng
    cout<<"Tong cac so tu nhien nho hon "<<n
                                     << " la: "<< S;
}
int sum(int n)             //Định nghĩa hàm sum
{
    int i, S =0;
    for (i=0; i<=n; i++)
        S = S+i;
    return S;
}
```

Một số qui tắc khi viết chương trình C

- Mỗi câu lệnh có thể viết trên một hay nhiều dòng nhưng phải được kết thúc bằng dấu “;”.
- Chú thích có thể được viết trên một dòng, nhiều dòng hoặc trên phần còn lại của câu lệnh.
- Khi sử dụng một hàm thư viện cần khai báo hàm ở đầu chương trình bằng cách toán tử `#include`, ví dụ: `#include<iostream.h>`
- Một chương trình chỉ có một hàm chính (main), có thể có thêm vài hàm khác (gọi là hàm con).

5. Các thành phần của chương trình C/C++ đơn giản (1)

(1) `#include <??>`

yêu cầu trình biên dịch đọc tập tin chứa các khai báo như khai báo hàm thư viện mà chương trình dùng.

VD: `iostream.h` → `cout`, `cin`, ...

`stdio.h` → `printf`, `scanf`, ...

`math.h` → `sqrt`, `sin`, `log`, `pow`, ...

`conio.h` → `getch`, `clrscr`, ...

5. Các thành phần của chương trình C/C++ đơn giản (2)

(2) Hàm chính, là thành phần buộc phải có trong mọi chương trình C.

Dạng đơn giản:

```
void main() { }
```

```
int main() { return 0; }
```

```
main() { return 0; }
```

5. Các thành phần của chương trình C/C++ đơn giản (3)

(3) Định nghĩa dữ liệu và các phát biểu.

- Các *phát biểu* là phần thực thi của chương trình.
(*đọc từ bàn phím, xuất ra màn hình, thực hiện tính toán, gọi hàm,...*)
- Các phát biểu được đặt giữa cặp ngoặc { và } của hàm (main), tạo nên “*thân hàm*”.
- Mỗi *phát biểu đơn* (*câu lệnh*) được kết thúc bởi ‘;’
- Các phát biểu cùng được đặt giữa { và } tạo thành *phát biểu ghép* (còn gọi *khối lệnh*).

5. Các thành phần của chương trình C/C++ đơn giản (4)

(4) Khai báo hàm và định nghĩa hàm.

Khai báo hàm là đưa ra một “*mẫu hàm*”, gồm tên và các tham số của hàm (*kết thúc bởi ;*).

```
int sum(int n) ;
```

Định nghĩa hàm gồm tên hàm, các tham số và thân hàm (chứa các phát biểu chương trình), thực thi một việc cụ thể.

5. Các thành phần của chương trình C/C++ đơn giản (5)

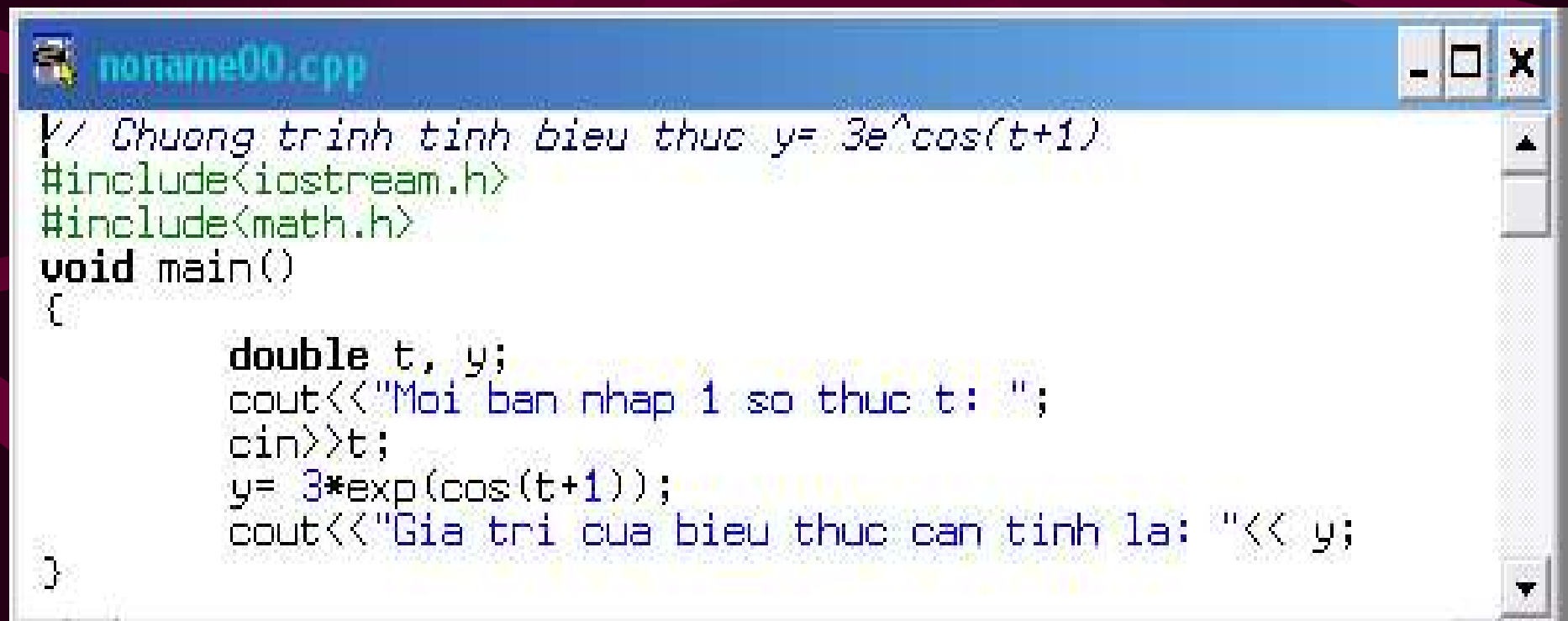
(5) Các chú thích, được trình biên dịch “bỏ qua”, không ảnh hưởng đến việc thực thi của chương trình.

Có hai loại chú thích:

- *Chú thích khối*, chú thích là phần văn bản đặt giữa */** và **/*
- *Chú thích dòng*, chú thích là phần văn bản đặt ngay sau cặp kí tự: *//*

Xuất dữ liệu (1)

- **cout** là đối tượng xuất chuẩn, xuất dữ liệu ra màn hình.
- Một phát biểu xuất kết quả ra màn hình, bao gồm: **cout**, phép toán xuất <<, *đối tượng* được xuất, và ';'.



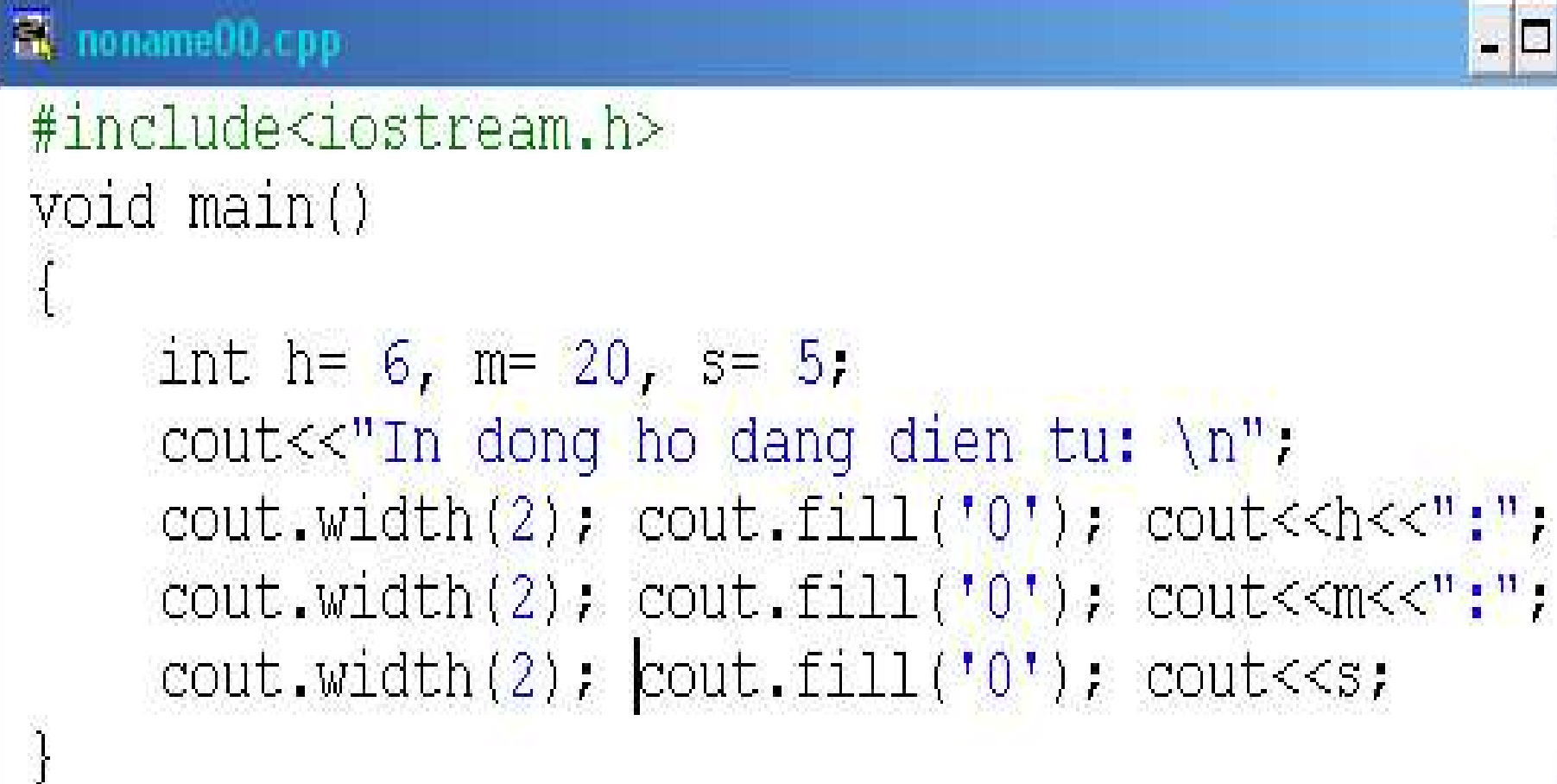
```
// Chương trình tính biểu thức  $y = 3e^{\cos(t+1)}$ 
#include<iostream.h>
#include<math.h>
void main()
{
    double t, y;
    cout<<"Moi ban nhap 1 so thuc t: ";
    cin>>t;
    y= 3*exp(cos(t+1));
    cout<<"Gia tri cua bieu thuc can tinh la: "<< y;
}
```

Xuất dữ liệu (2)

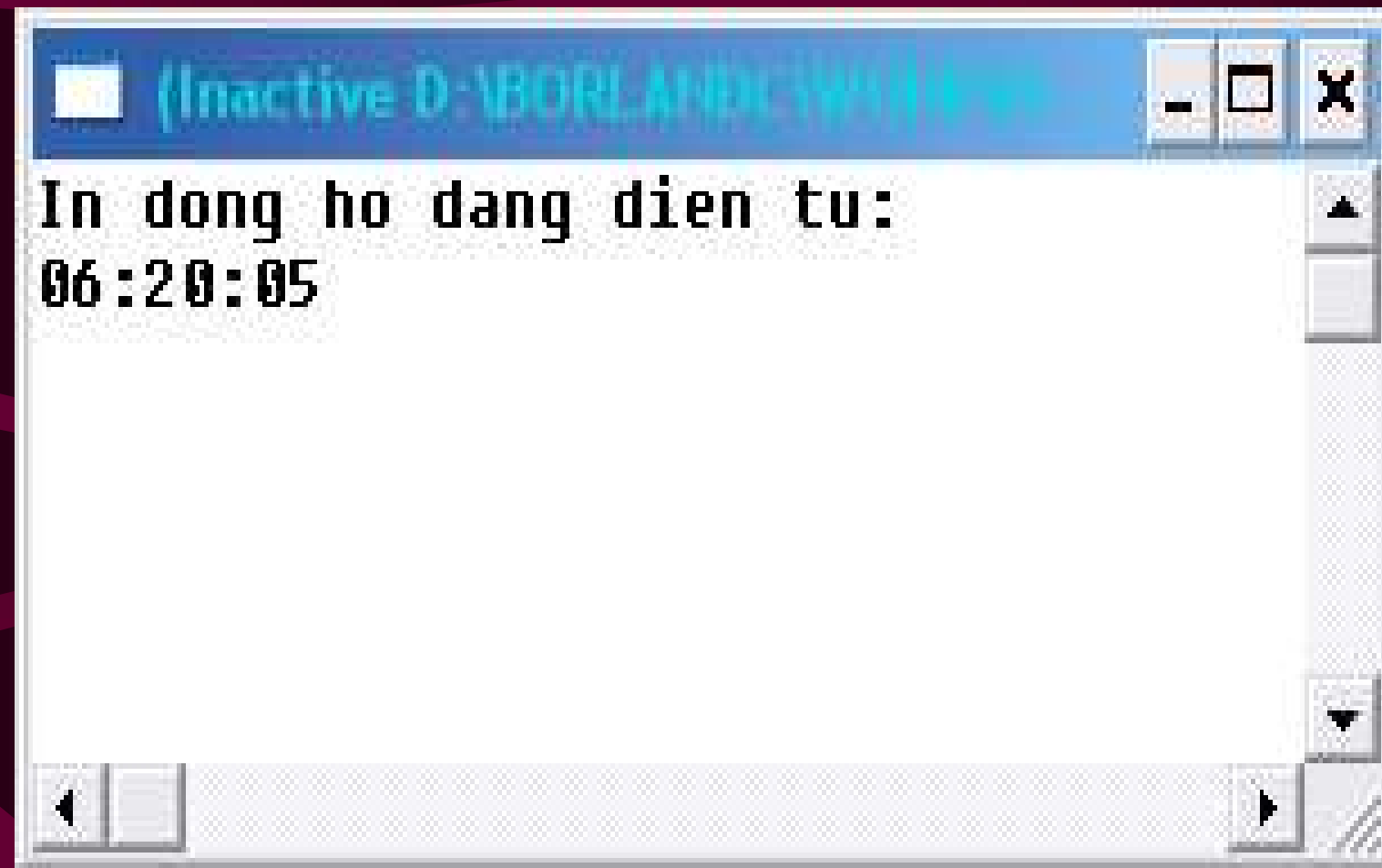
Một đối tượng được xuất có thể là:

- Số nguyên (số thực) hay biến nguyên (biến thực).
- Kí tự (một hằng kí tự được đặt giữa cặp dấu ‘ ’) hoặc biến kiểu kí tự.
- Thông điệp gồm nhiều kí tự (*chuỗi kí tự*) được đặt giữa cặp dấu “ ”.

Định dạng xuất, như: cho phép có bao nhiêu số ở phần thập phân, canh lề phải dữ liệu xuất...



```
#include<iostream.h>
void main()
{
    int h= 6, m= 20, s= 5;
    cout<<"In dong ho dang dien tu: \n";
    cout.width(2); cout.fill('0'); cout<<h<<": ";
    cout.width(2); cout.fill('0'); cout<<m<<": ";
    cout.width(2); cout.fill('0'); cout<<s;
}
```



Xuất dữ liệu (3)

- Nhiều phát biểu xuất có thể được nối lại thành một phát biểu xuất, khi đó cần lưu ý là trước mỗi đối tượng được xuất là một phép toán xuất và ngược lại.
- Các đối tượng được xuất ra sẽ liên tiếp nhau trên cùng một dòng.
- Nếu muốn một đối tượng được xuất ra trên một dòng mới thì gọi endl hay “\n” trong phát biểu xuất.

Nhập dữ liệu

- **cin** là dòng nhập chuẩn, đọc dữ liệu được gõ từ bàn phím.
- Dạng tổng quát: `cin>> var;`
var là một biến nào đó
- Phát biểu nhập có nhiều đối tượng:
`cin>>var1>>var2...>>varN`