



# IT001 - NHẬP MÔN LẬP TRÌNH

## CHƯƠNG 3.3: NHẬP XUẤT TRONG C++

Nhập xuất dữ liệu là hai thao tác cơ bản trong lập trình, cho phép chương trình tương tác với người dùng. Chương này sẽ giới thiệu về các hàm trong C++ hỗ trợ nhập xuất dữ liệu từ bàn phím và xử lý lỗi nhập dữ liệu.

Khoa Khoa học Máy tính



# NỘI DUNG

- 3.1 Cấu trúc chương trình C++
- 3.2 Bộ từ vựng (**keywords**) trong C++
- 3.3 Các kiểu dữ liệu cơ sở (**Fundamental data types**)
- 3.4 Biến (**Variable**)
- 3.5 Hằng (**Constant**)
- 3.6 Các phép toán (**Operator**)
- 3.7 Biểu thức và độ ưu tiên toán tử
- 3.8 Nhập xuất dữ liệu
- Bài tập



## 3.8 Nhập xuất dữ liệu



# NỘI DUNG

- 3.8.1 Thư viện nhập xuất
- 3.8.2 Câu lệnh xuất
- 3.8.3 Câu lệnh nhập
- 3.8.4 Hàm xóa bộ nhớ đệm
- Bài tập



## 3.8.1 Thư viện nhập xuất

- Có 2 loại thư viện nhập xuất trong C++
  - Thư viện nhập xuất `<stdio.h>` (kế thừa từ ngôn ngữ C)
  - Thư viện nhập xuất tiêu chuẩn `<iostream>` của C++.

	<b>Ngôn ngữ C++</b>	<b>Ngôn ngữ C</b>
Khai báo thư viện và namespace	<code>#include &lt;iostream&gt;</code> <code>using namespace std;</code>	<code>#include &lt;stdio.h&gt;</code>
Lệnh nhập	<code>cin &gt;&gt;</code> <code>cin.get, cin.getline,</code> <code>getline (&lt;string&gt;)</code>	<code>scanf</code> <code>fgetc, getc, getchar</code> <code>fgets</code>
Lệnh xuất	<code>cout &lt;&lt;</code>	<code>printf</code> <code>putchar, putc</code> <code>puts, fputs</code>



## 3.8.2 Câu lệnh xuất

- Lệnh xuất **cout <<** (C++)
- Hàm **printf** (C)
- Các hàm xuất khác



# Câu lệnh xuất: cout

- Thư viện: `#include <iostream>`
- Cú pháp: `std::cout << Tham_số_1 << Tham_số_2 << ... << Tham_số_k;`
- Tham số có thể:
  - Chuỗi hằng
  - Ký tự điều khiển (escape sequence)
  - Biến, hằng số, biểu thức, hàm
- Ví dụ: `int i=10;  
std::cout << "Chuong trinh xuat gia tri i \n";  
std::cout << "gia tri la " << i;`

C++ sẽ tự động nhận dạng  
kiểu dữ liệu và xuất dữ liệu ra  
màn hình một cách phù hợp.

Chuong trinh xuat gia tri i  
gia tri la 10



# Câu lệnh xuất: cout

- Ví dụ:

```
#include <iostream>

using namespace std;
int main () {
    int a=0;
    float b=5.5;
    char c='c';
    string str = "Nhập Môn Lập Trình";
    char s[] = "Học Tốt Thi Tốt";
    std::cout << a << " và " << b << " và " << c << std::endl;
    std::cout << str << " & " << s << std::endl;
    return 0;
}
```

**Thực thi chương trình:**

0 và 5.5 và c

Nhập Môn Lập Trình & Học Tốt Thi Tốt



# Câu lệnh xuất: cout

- Ký tự điều khiển (escape sequence): Gồm dấu `\'` và một ký tự như trong bảng sau:

Escape sequence	Miêu tả	Escape sequence	Miêu tả
\'	single quote – Dấu nháy đơn	\r	carriage return - Trở về đầu dòng
\\"	double quote – Dấu nháy kép	\t	horizontal tab - Tab ngang
\?	question mark – Ký tự dấu hỏi	\v	vertical tab - Tab dọc
\\\	Backslash - Ký tự backslash	\nnn	arbitrary octal value - mã bát phân
\a	audible bell - Phát ra âm thanh báo	\xnn	arbitrary hexadecimal value – mã thập lục phân
\b	Backspace – Lùi một ký tự	\unnnnn	universal character name (arbitrary <u>Unicode</u> value); may result in several characters
\f	Form feed – Chuyển trang mới		
\n	new line - Xuống dòng		



# Ví dụ: Ký tự điều khiển \n \t \r

```
#include <iostream>

int main() {
    std::cout << "'1'\t'2'\t'3'\n";
    std::cout << "Hom nay\troi nang dep\n\n";
    std::cout << "Chao lop, \"Nhap mon lap trinh\"\n";
    std::cout << "Chuc cac em \rhoc tot";
}
```

Kết quả thực thi:

'1' '2' '3'

Hom nay

troi nang dep

Chao lop, "Nhap mon lap trinh"  
hoc totc em



# Ví dụ: Ký tự escape \nnn

- Ký tự escape \nnn dùng để biểu diễn một ký tự thông qua mã số bát phân (octal). nnn là một chuỗi từ một đến ba chữ số bát phân (octal digits) nằm trong khoảng từ 000 đến 377 (tương đương với giá trị thập phân từ 0 đến 255).

```
#include <iostream>
```

```
int main() {
```

```
    char c = '\104';
```

```
    std::cout << "Ký tu: " << c << std::endl;
```

Ta có:

$$(104)_8 = 1*8^2 + 0*8^1 + 4*8^0 = (68)_{10}$$

```
    std::cout << "Các ký tự: \144 \166" << std::endl;
```

```
    return 0;
```

```
}
```

Kết quả thực thi:

Ký tu: D

Các ký tự: d v



# Ví dụ: Chuyển màu văn bản

```
#include <iostream>
```

```
int main() {
    // In văn bản màu đỏ
    std::cout << "\x1B[31m" << "Đây là văn bản màu đỏ" << "\x1B[0m" << std::endl;
```

```
    return 0;
}
```

Kết quả thực thi:

Đây là văn bản màu đỏ.

Tham khảo thêm tại: <https://www.geeksforgeeks.org/how-to-print-colored-text-in-c/>



# Một số thiết lập của cout

- Thiết lập độ rộng

Cú pháp: **cout.width(n)** - Với n là độ rộng mới

Chú ý: Độ rộng quy định n chỉ có tác dụng cho một giá trị xuất.  
Sau đó C++ lại áp dụng độ rộng quy định bằng 0.

Ví dụ:

```
#include <iostream>
int main() {
    int a = 1706;
    std::cout << a << "\n";
    std::cout.width(10);
    std::cout << a;
    return 0;
}
```

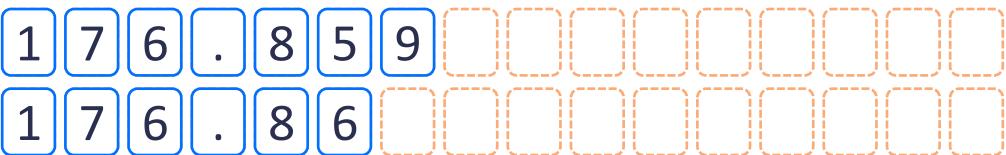




# Một số thiết lập của cout

- Độ chính xác số thực:
- Cú pháp: **cout.precision(n)** - Với n là độ chính xác áp dụng
- Chú ý: Độ chính xác được thiết lập sẽ có hiệu lực cho tới khi gặp một câu lệnh thiết lập độ chính xác mới
- Ví dụ:

```
#include <iostream>
int main() {
    float x = 176.859;
    std::cout << x << "\n";
    std::cout.precision(5);
    std::cout << x << "\n";
    return 0;
}
```





# Một số hàm thiết lập hữu ích khác

Thư viện	Ví dụ
<iostream>	<pre>std::boolalpha std::noboolalpha  std::cin.setf(std::ios::boolalpha); std::cout.setf(std::ios::boolalpha); std::cout.unsetf(std::ios::boolalpha); std::cin.unsetf(std::ios::boolalpha);  std::hex, std::oct, std::scientific  std::showbase, std::noshowbase  std::nouppcase, std::uppercase  std::fixed  std::internal, std::left, std::right</pre>
<iomanip>	<pre>std::setw  std::setprecision  std::setfill  std::setbase  std::setiosflags, std::resetiosflags</pre>



# Ví dụ: Hiển thị theo định dạng số thập lục phân

```
#include <iostream>
#include <iomanip>

int main() {
    int value = 128;

    std::cout << "Decimal: " << value << std::endl;
    std::cout << "Hexadecimal: 0x" << std::setfill('0')
        << std::setw(4) << std::hex << value << std::endl;
    std::cout << "Hexadecimal: 0x" << std::uppercase
        << std::hex << value << std::endl;

    return 0;
}
```

## Kết quả thực thi:

```
Decimal: 128
Hexadecimal: 0x0080
Hexadecimal: 0x80
```



# Ví dụ: Hiển thị số theo định dạng showbase

```
#include <iostream>
#include <iomanip> // Thư viện để sử dụng showbase và các thao tác định dạng khác
using namespace std;
int main() {
    int value = 255;

    cout << "Without showbase:" << endl;
    cout << "\tDecimal: " << dec << value << endl;
    cout << "\tHexadecimal: " << hex << value << endl;
    cout << "\tOctal: " << oct << value << endl;

    cout << "With showbase:" << endl;
    cout << showbase;
    cout << "\tDecimal: " << dec << value << endl;
    cout << "\tHexadecimal: " << hex << value << endl;
    cout << "\tOctal: " << oct << value << endl;

    cout << "Turning off showbase:" << endl;
    cout << noshowbase;
    cout << "\tDecimal: " << dec << value << endl;
    cout << "\tHexadecimal: " << hex << value << endl;
    cout << "\tOctal: " << oct << value << endl;

    return 0;
}
```

## Kết quả thực thi:

Without showbase:

Decimal: 255

Hexadecimal: ff

Octal: 377

With showbase:

Decimal: 255

Hexadecimal: 0xff

Octal: 0377

Turning off showbase:

Decimal: 255

Hexadecimal: ff

Octal: 377



# Ví dụ: Xuất giá trị với fixed và setprecision

```
#include <iostream>
#include <iomanip> // Thư viện để sử dụng fixed và setprecision
using namespace std;

int main() {
    float value = 123.456789;

    cout << "Without fixed: " << value << endl;

    cout << "With fixed: " << fixed << value << endl;

    cout << "With fixed and setprecision(3): "
        << fixed << setprecision(3) << value << endl;

    cout << "With fixed and setprecision(5): "
        << fixed << setprecision(4) << value << endl;

    return 0;
}
```

## Kết quả thực thi:

Without fixed: 123.457

With fixed: 123.456787

With fixed and setprecision(3): 123.457

With fixed and setprecision(5): 123.4568



# Ví dụ: Hiển thị giá trị boolean

```
#include <iostream>
using namespace std;

int main() {
    bool flagTrue = true;
    bool flagFalse = false;

    cout << "Without boolalpha: " << flagTrue << " " << flagFalse << endl;

    cout << boolalpha; // Kích hoạt boolalpha
    cout << "With boolalpha: " << flagTrue << " " << flagFalse << endl;

    cout << noboolalpha; // Tắt boolalpha
    cout << "Turning off boolalpha: " << flagTrue << " " << flagFalse << endl;
    return 0;
}
```

## Kết quả thực thi:

```
Without boolalpha: 1 0
With boolalpha: true false
Turning off boolalpha: 1 0
```



# Câu lệnh xuất printf

- **Thư viện:** #include <stdio.h> (standard input/output)
- **Cú pháp:**

```
int printf(const char *format, <[Danh sách đối số]>);
```

- **Trong đó:**
  - **format:** Một chuỗi định dạng chứa văn bản và các ký tự định dạng. Các ký tự định dạng bắt đầu bằng dấu phần trăm (%) và xác định kiểu dữ liệu của đối số tương ứng.
  - **<[Danh sách đối số]>:** Một số lượng không xác định các đối số, mỗi đối số tương ứng với một ký tự định dạng trong chuỗi format.



# Chuỗi định dạng

- **Chuỗi định dạng** trong hàm **printf** C++ là một phần quan trọng để kiểm soát cách thức hiển thị dữ liệu ra màn hình. Chuỗi này bao gồm các ký tự đặc biệt và các phần tử định dạng để tạo ra bố cục và định dạng mong muốn cho dữ liệu.
- **Cú pháp:**

```
%[flags][width][.precision][length]specifier
```



# Specifier (Ký tự định dạng)

`%[flags][width][.precision][length]specifier`

- Bao gồm các ký tự định dạng sau:
  - `d`: Định dạng cho số nguyên (integer)
  - `f`: Định dạng cho số thực (float)
  - `c`: Định dạng cho ký tự (character)
  - `s`: Định dạng cho chuỗi (string)
  - `p`: Định dạng cho địa chỉ (pointer), ...

- Ví dụ 1:

```
int a = 10, b = 20;
```

```
printf("Gia tri a la %d, b la %d", a, b);
```

→ Xuất ra **Gia tri a la 10, b la 20**

- Ví dụ 2:

```
float x = 15.06;
```

```
printf("x = %f", x);
```

→ Xuất ra **x = 15.060000**



# Flags (Cờ)

```
%[flags][width][.precision][length]specifier
```

- **Bao gồm các ký tự cờ sau:**

- ` - ` : Căn lề trái (left-justify)
- ` + ` : Buộc in dấu + trước các số dương
- (space) : Chèn khoảng trắng nếu giá trị không có dấu (space for positive values)
- ` # ` : Thay đổi cách hiển thị (ví dụ: thêm 0x cho số thập lục phân)
- ` 0 ` : Điene số 0 phía trước cho đủ độ rộng (zero-padding)

- Ví dụ:

```
int num = 42;  
double pi = 3.14;  
// In số nguyên ở dạng thập lục phân, thêm tiền tố 0x  
printf("%#x\n", num); // Output: "0x2a"  
// In số thực với dấu chấm thập phân luôn xuất hiện  
printf("%#f\n", pi); // Output: "3.140000"
```

```
int num = 50;  
// In số nguyên với độ rộng tối thiểu 4, căn lề trái  
printf("|%-4d|\n", num); // Output: "|50 |"
```



# Width (Độ rộng)

```
%[flags][width][.precision][length]specifier
```

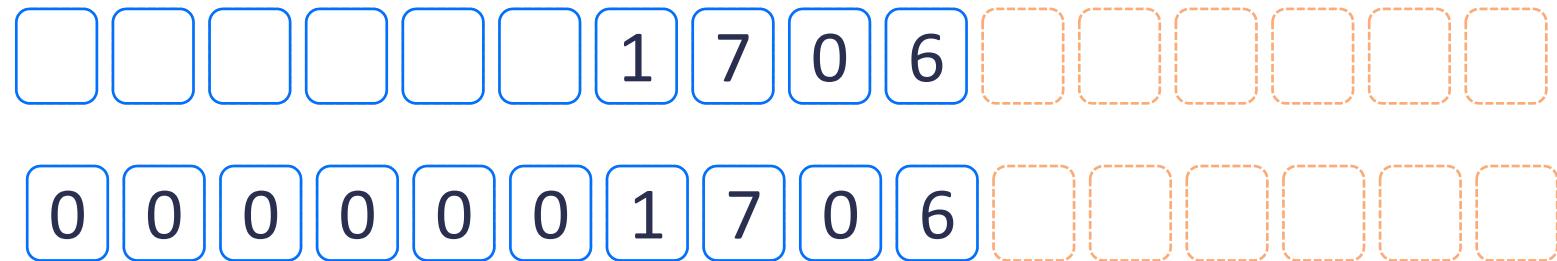
- **Width (Độ rộng):** Số nguyên chỉ độ rộng tối thiểu của giá trị in ra.
- Ví dụ:

```
int a = 1706;
```

```
printf("%10d\n", a);
```

```
printf("%010d", a);
```

→ Kết quả:





# Precision (Độ chính xác)

```
%[flags][width][.precision][length]specifier
```

- **Precision (Độ chính xác):** được chỉ định sau dấu chấm `.`.
  - Đối số là số thực: nó chỉ định số chữ số thập phân
  - Đối số là chuỗi: nó chỉ định số ký tự tối đa của chuỗi.
- Ví dụ:

```
int a = 176.8512;  
printf("%.2d", a); // Output: "176.85"
```

```
char str[] = "LapTrinh";  
// In chuỗi với độ chính xác 5 (chỉ in 5 ký tự đầu)  
printf("%.5s\n", str); // Output: "LapTr"
```



# Ví dụ

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main() {
    int a = 1, b = 2;
    printf("%d", a); // Xuất giá trị của biến a
    printf(" cong "); // Xuất chuỗi " cong "
    printf("%d", b); // Xuất giá trị của biến b
    printf(" bang "); // Xuất chuỗi " bang "
    printf("%d", a + b); // Xuất giá trị của a + b
    printf("\n"); // Đặc tả điều khiển xuống dòng
    printf("%d cong %d bang %d\n", a, b, a+b);
    return 0;
}
```

## Thực thi chương trình:

```
1 cong 2 bang 3
1 cong 2 bang 3
```



# Ví dụ:

```
#include <stdio.h>

int main() {
    printf ("Characters: %c %c \n", 'a', 65);
    printf ("Decimals: %d %ld\n", 1977, 650000L);
    printf ("Preceding with blanks: %10d \n", 1977);
    printf ("Preceding with zeros: %010d \n", 1977);
    printf ("Some different radices: %d %x %o %#x %#o \n", 100, 100, 100, 100,
100);
    printf ("floats: %4.2f %+ .0e %E \n", 3.1416, 3.1416, 3.1416);
    printf ("Width trick: %*d \n", 5, 10);
    printf ("%s \n", "A string");
    return 0;
}
```

## Thực thi chương trình:

Characters: a A

Decimals: 1977 650000

Preceding with blanks: 1977

Preceding with zeros: 0000001977

Some different radices: 100 64 144 0x64 0144

floats: 3.14 +3e+000 3.141600E+000

Width trick: 10

A string



# Một số lệnh xuất khác

- Ví dụ:

```
#include <stdio.h>  
  
int main () {  
  
    char str[] = "Hello world!";  
  
    puts(str);  
  
    puts(str);  
  
    fputs(str, stdout);  
  
    fputs(str, stdout);  
  
    printf ("%s", str);  
  
    printf ("%s", str);  
  
}
```

Thực thi chương trình

Hello world!

Hello world!

Hello world!Hello world!Hello world!Hello world!



### 3.8.3 Câu lệnh nhập

- Câu lệnh nhập cin >> (C++)
- Câu lệnh nhập scanf ( C )
- Các lệnh nhập khác:
  - Nhập ký tự: fgetc, getc, getchar, cin.get
  - Nhập chuỗi C-string: fgets, cin.get, cin.getline
  - Nhập chuỗi string: getline



# Câu lệnh nhập: std::cin>> (C++)

- **cin** là một đối tượng được định nghĩa trong thư viện **iostream** thuộc namespace std trong C++. Đối tượng này kết hợp với toán tử **>>** được sử dụng để đọc dữ liệu từ thiết bị nhập chuẩn (mặc định là bàn phím).

- Thư viện: **#include <iostream>**

- Cú pháp: **std::cin >> Tham\_số\_1 >> Tham\_số\_2 >> ... >> Tham\_số\_k;**

- Ví dụ:

```
int namsinh;  
std::cin >> namsinh;
```



# Câu lệnh nhập: std::cin>> (C++)

- Có thể nhập dữ liệu cho các biến có kiểu dữ liệu cơ sở **khác nhau** trên cùng một hàm cin (bool, float, double, ...).
- ➔ Khi đó, các biến sẽ được phân cách bằng một ký tự trắng (white space characters). White space character bao gồm:
  - Ký tự khoảng trắng (ASCII code = 32)
  - Ký tự tab ('\t')
  - Ký tự xuống dòng ('\n')



# Câu lệnh nhập: std::cin>> (C++)

Ví dụ: Kết hợp Nhập nhiều giá trị có kiểu khác nhau trên cùng dòng cin>>

```
#include <iostream>
using namespace std;
int main() {
    char c;
    bool b;
    int i;
    double d;
    cout << "Nhap: " << endl;
    cin >> c >> b >> i >> d;
    cout << "Xuat: " << endl;
    cout << c << '\t' << b << '\t' << i << '\t' << d;
    return 0;
}
```

## Thực thi chương trình:

Nhap:

D 1

8 9.5

Xuat:

D 1 8 9.5



# Câu lệnh nhập: scanf (Ngôn ngữ C)

- **Thư viện:** #include <stdio.h>

- **Cú pháp:**

```
int scanf ( const char * format, ... (additional arguments));
```

- **Trong đó:**

- Format: giống định dạng xuất nhưng chỉ có các đặc tả

- (additional arguments): Các đối số là tên các biến sẽ chứa giá trị nhập và  
được đặt trước dấu &



# Câu lệnh nhập: scanf (Ngôn ngữ C)

- Ví dụ:

```
scanf("%d", &a); // Nhập giá trị cho biến số nguyên a
```

```
scanf("%d", &b); // Nhập giá trị cho biến số nguyên b
```

```
scanf("%d%d", &a, &b);
```

- Các câu lệnh sau đây sai

```
scanf("%d", a); // Thiếu dấu &
```

```
scanf("%d", &a, &b); // Thiếu %d cho biến b
```

```
scanf("%f", &a); // a là biến kiểu số nguyên
```

```
scanf("%9d", &a); // không được định dạng
```

```
scanf("a = %d, b = %d", &a, &b);
```



# Các hàm hỗ trợ nhập khác

- Các lệnh nhập khác:
  - Nhập ký tự: fgetc, getc, getchar, `cin.get` (C++)
  - Nhập chuỗi C-string: fgets, `cin.get` (C++), `cin.getline` (C++)
  - Nhập chuỗi string (<string> header): `std::getline` (C++)



# Nhập ký tự: cin.get

- **Cú pháp:** `istream& get (char& c);`
- **Ý nghĩa:**
  - Trích xuất một ký tự đơn lẻ từ Stream. Ký tự này có thể được trả về hoặc được gán làm giá trị cho đối số tham chiếu của nó.
- **Ví dụ:**

```
#include <iostream>
using namespace std;
int main() {
    char c;
    cout << "Nhập Lan 1: ";    cin.get(c);
    cout << "Xuất: " << c << endl;
    cin.ignore();
    cout << "Nhập Lan 2: ";    c=cin.get();
    cout << "Xuất: " << c << endl;
    return 0;
}
```

## Kết quả thực thi:

```
Nhập Lan 1: a
Xuất: a
Nhập Lan 2: b
Xuất: b
```



# Nhập ký tự: Lệnh getchar

- Cú pháp:

```
int getchar ( void );
```

- Trong thư viện `<stdio.h>`
- Hàm trả về ký tự tiếp theo từ đầu vào tiêu chuẩn (standard input - `stdin`).
- `getchar` tương đương với việc gọi `getc` với `stdin` làm đối số.
- Ví dụ:

```
#include <stdio.h>
int main() {
    char c;
    printf("Nhap: ");
    c=getchar();
    printf("Xuat: "); putchar(c);
    return 0;
}
```

Thực thi chương trình:

Nhap: Lap Trinh  
Xuat: L



# Nhập ký tự: Lệnh fgetc và getc

- Cú pháp:

```
int fgetc ( FILE * stream );  
int getc ( FILE * stream );
```

- Trong thư viện `<stdio.h>`
- Hàm `fgetc`, `getc` có thể đọc các ký tự từ đầu vào stream (con trỏ đến đối tượng `FILE` đầu vào) và trả về ký tự đọc.
- `stdin` được sử dụng như là con trỏ file stream để đọc đầu vào tiêu chuẩn từ bàn phím.



# Nhập ký tự: Lệnh fgetc và getc

- Ví dụ:

```
#include <stdio.h>

int main() {
    char c1, c2;
    printf("Nhập: "); c1=getchar(stdin);
    printf("Xuất: %c", c1);
    fflush(stdin);
    printf("\nNhập: "); c2=fgetc(stdin);
    printf("Xuất: %c", c2);
    return 0;
}
```

## Thực thi chương trình:

Nhap: Nhập M  
Xuat: N  
Nhập: Lập Trình  
Xuất: L



# Ví dụ: Tổng hợp các cách nhập ký tự

```
#include <iostream>
using namespace std;
int main() {
    char c;
    cout << "Nhập dung cin: "; cin >> c;
    cout << "Xuất: " << c << endl;
    cin.ignore();
    cout << "Nhập dung cin.get(): "; c=cin.get();
    cout << "Xuất: " << c << endl;
    cin.ignore();
    cout << "Nhập dung getchar(): "; c=getchar();
    cout << "Xuất: " << c << endl;
    cin.ignore();
    cout << "Nhập dung c=getc(stdin): "; c=getc(stdin);
    cout << "Xuất: " << c << endl;
    cin.ignore();
    cout << "Nhập dung fgetc(stdin): "; c=fgetc(stdin);
    cout << "Xuất: " << c << endl;
    return 0;
}
```

## Kết quả thực thi:

```
Nhập dung cin: a
Xuất: a
Nhập dung cin.get(): b
Xuất: b
Nhập dung getchar(): c
Xuất: c
Nhập dung c=getc(stdin): d
Xuất: d
Nhập dung fgetc(stdin): e
Xuất: e
```



# Nhập chuỗi

- Nhập chuỗi bằng cin:

```
std::string mystring;
```

```
std::cin >> mystring;
```

- ➔ “`std::cin >>`” xem khoảng trắng (khoảng trắng, tab, enter) là kết thúc giá trị được nhập.
- ➔ Nếu trong dòng nhập tồn tại khoảng trắng thì **cin chỉ đọc được đến khoảng trắng** (khoảng trắng không được nhập vào chuỗi).
- ➔ Dùng hàm `std::getline` trong thư viện `<string>` hoặc `fgets` (giới thiệu), `cin.get` (giới thiệu), `cin.getline` để nhập C-string



# Nhập C-String: Hàm fgets

- Cú pháp: `char * fgets ( char * str, int num, FILE * stream );`
- Hàm fgets đọc các ký tự từ stream (con trỏ trả về đối tượng FILE đầu vào) và lưu trữ dưới dạng C-String thành str cho đến khi (num-1) ký tự được đọc hoặc đến dòng mới \n hoặc cuối tệp stream (tùy điều kiện nào xảy ra trước).
- Nếu ký tự dòng mới \n làm cho hàm fgets ngừng → ký tự \n sẽ được đưa vào chuỗi str.
- Ký tự kết thúc chuỗi \0 sẽ được tự động thêm vào str.
- **stdin** được sử dụng như là con trỏ file stream để đọc đầu vào tiêu chuẩn từ bàn phím.
- Chèn header `<stdio.h>` để dùng hàm fgets.



# Nhập C-String: Hàm fgets

- Ví dụ:

```
#include <iostream>
using namespace std;

int main () {
    char str[9];
    printf("Nhập lần 1: ");    fgets (str, 9, stdin);
    printf("Xuất: %s!\n", str);

    printf("Nhập lần 2: ");    fgets (str, 9, stdin);
    printf("Xuất: %s!\n", str);

    printf("Nhập lần 3: ");    fgets (str, 9, stdin);
    printf("Xuất: %s!\n", str);
    return 0;
}
```

Lưu ý: Nếu ký tự dòng mới \n làm cho hàm fgets ngừng → ký tự \n sẽ được đưa vào chuỗi str (xem ví dụ Nhập lần 1 và lần 3)

## Thực thi chương trình:

```
Nhap lan 1: 12345
Xuat: 12345
!
Nhap lan 2: 1234567890
Xuat: 12345678!
Nhap lan 3: Xuat: 90
!
```



# Nhập C-String: Lệnh cin.getline

- Cú pháp:

(1) `istream& getline (char* str, streamsize num);`

(2) `istream& getline (char* str, streamsize num, char delim );`

- Đọc các ký tự từ Stream đầu vào và lưu trữ vào str dưới dạng C-String, cho đến khi đọc được ký tự delim hoặc đọc hết (num-1) ký tự. Ký tự null '\0' được tự động thêm vào str.
- Ký tự delim: (cú pháp (1) mặc định delim là '\n'): **delim được trích xuất khỏi chuỗi Stream khi được tìm thấy, và không được ghi vào str.**
- Nếu hàm dừng đọc vì đọc hết num-1 mà không tìm thấy ký tự delim, sẽ bị lỗi failbit.
- Hàm này bị overload đối với các đối tượng string trong header <string>.

Xem: `getline (string)`



# Nhập C-String: Lệnh cin.getline

- Ví dụ:

```
#include <iostream>
using namespace std;
int main () {
    char str[10];
    cout << "Nhập lần 1: "    cin.getline (str, 10);
    cout << "Xuất: " << str << "!" << endl;

    cout << "Nhập lần 2: "    cin.getline (str, 10, 'r');
    cout << "Xuất: " << str << "!" << endl;

    cout << "Nhập lần 3: "    cin.getline (str, 10);
    cout << "Xuất: " << str << "!" << endl;
    return 0;
}
```

## Thực thi chương trình:

Nhap lan 1: Nhap Mon  
Xuat: Nhap Mon!  
Nhap lan 2: Lap Trinh  
Xuat: Lap T!  
Nhap lan 3: Xuat: inh!

Lưu ý: ký tự delim không được đọc cho lần nhập kế tiếp



# Nhập C-String: Lệnh cin.getline

- Ví dụ:

```
#include <iostream>
#include <limits>
using namespace std;
int main () {
    char str[9];
    cout << "Nhập lần 1: "; cin.getline (str, 9, '0');
    if(cin.fail()) cout << "Fail.\n";
    else cout << "Xuất: " << str << "!" << endl;

    cout << "Nhập lần 2: "; cin.getline (str, 9);
    cout << "Xuất: " << str << "!" << endl;
    return 0;
}
```

## Lưu ý:

- Hàm đọc không bị lỗi, do ký tự '0' nằm ngay sau số lượng num-1 ký tự nhập vào
- Ký tự '0' sẽ không được đọc vào str
- Vậy trên dòng Stream còn 12345\n
- Lần nhập thứ 2 thì str sẽ đọc 12345

## Thực thi chương trình:

```
Nhập lần 1: 12345678012345
Xuất: 12345678
Nhập lần 2: Xuất: 12345
```



# Nhập C-String: Lệnh cin.getline

- Ví dụ:

```
#include <iostream>
#include <limits>
using namespace std;
int main () {
    char str[9];
    cout << "Nhập lần 1: "; cin.getline (str, 9, '0');
    if(cin.fail()) cout << "Fail.\n";
    else cout << "Xuat: " << str << "!" << endl;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Nhập lần 2: "; cin.getline (str, 9);
    cout << "Xuat: " << str << "!" << endl;
    return 0;
}
```

## Thực thi chương trình:

```
Nhập lần 1: 1234567890123
Fail.
Nhập lần 2: Xuat: !
```

Lưu ý: Nếu hàm `cin.getline` dừng vì đọc hết (`num-1)=8` ký tự mà không tìm thấy ký tự `delim` → Lỗi `failbit`.



# Nhập C-String: Lệnh cin.get

- Cú pháp:

```
istream& get (char* str, streamsize n);
```

```
istream& get (char* str, streamsize n, char delim);
```

- Trích xuất các ký tự từ dòng Stream và lưu trong str dưới dạng C-string, cho đến khi (n-1) ký tự được đọc hoặc gặp ký tự delim.
- Ký tự phân cách **không được trích xuất** khỏi chuỗi đầu vào nếu được tìm thấy và vẫn ở đó dưới dạng ký tự tiếp theo được trích xuất từ stream (xem getline để biết giải pháp thay thế loại bỏ ký tự delim).
- Ký tự null ('\0') được tự động thêm vào chuỗi str nếu n>0, kể cả khi chuỗi trống được trích xuất.



# Nhập C-String: Lệnh cin.get

- Ví dụ:

```
#include <iostream>
using namespace std;
int main () {
    char str[10];
    cout << "Nhập lần 1: "    cin.get (str, 10);
    cout << "Xuất: " << str << "!" << endl;

    cout << "Nhập lần 2: "    cin.get (str, 10, 'r');
    cout << "Xuất: " << str << "!" << endl;

    cout << "Nhập lần 3: "    cin.get (str, 10);
    cout << "Xuất: " << str << "!" << endl;
    return 0;
}
```

## Thực thi chương trình:

Nhap lan 1: Nhap Mon  
Xuat: Nhap Mon!  
Nhap lan 2: Lap Trinh  
Xuat:  
Lap T!  
Nhap lan 3: Xuat: rinh!

Lưu ý: ký tự delim được đọc cho lần nhập kế tiếp



# Nhập chuỗi string: Hàm getline (string)

- Cú pháp:

```
(1) istream& getline (istream& is, string& str, char delim);
```

```
istream& getline (istream&& is, string& str, char delim);
```

```
(2) istream& getline (istream& is, string& str);
```

```
istream& getline (istream&& is, string& str);
```

- Chèn thư viện `<iostream>`, `<string>` và namespace std
- Trích xuất các ký tự từ `is` và lưu trữ chúng vào `str` cho đến khi tìm thấy dấu phân tách ký tự `delim` (hoặc ký tự dòng mới, '`\n`' → như cú pháp thứ (2)).
- Nếu tìm thấy ký tự `delim`, nó sẽ được trích xuất và loại bỏ (**tức là `delim` không được lưu trữ và thao tác nhập tiếp theo sẽ bắt đầu sau `delim`**).
- Lưu ý rằng mọi nội dung trong `str` trước lệnh gọi sẽ được thay thế bằng chuỗi mới được trích xuất.



# Nhập chuỗi string: Hàm getline (string)

- Ví dụ:

```
#include <iostream>
#include <string>
#include <limits>
using namespace std;
int main () {
    string str;
    cout << "Nhập lần 1: ";
    getline (cin, str, 'o'); // Nhập cho đến khi gặp ký tự 'o' thì kết thúc
    cout << "Xuất: " << str << endl;

    cout << "Nhập lần 2: ";
    getline (cin, str);
    cout << "Xuất: " << str;
    return 0;
}
```

## Thực thi chương trình:

```
Nhập lần 1: Nhập Mon
Xuất: Nhập M
Nhập lần 2: Xuất: n
```

Lưu ý: thao tác nhập lần 2 sẽ bắt đầu sau ký tự  
delim 'o'



# Nhập chuỗi string: Hàm getline (string)

- Ví dụ:

```
#include <iostream>
#include <string>
#include <limits>
using namespace std;
int main () {
    string str;
    cout << "Nhập lan 1: ";
    getline (cin, str, 'o'); // Nhập cho đến khi gặp ký tự 'o' thì kết thúc
    cout << "Xuất: " << str << endl;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Nhập lan 2: ";
    getline (cin, str);
    cout << "Xuất: " << str;
    return 0;
}
```

## Thực thi chương trình:

```
Nhập lan 1: Nhập Mon
Xuất: Nhập M
Nhập lan 2: Lập Trình
Xuất: Lập Trình
```

Lưu ý: `cin.ignore` dùng để xóa `numeric_limits<streamsize>::max()` trong stream hoặc cho đến khi gặp ký tự kết thúc dòng '\n'



# Ví dụ: Tổng hợp các hàm nhập xuất chuỗi

```
#include <stdio.h>
#include <iostream>
#include <string>
#include <limits>
using namespace std;
int main () {
    char str1[100], str2[100], str3[100], str4[100]; string str5;
    cout << "Nhập str1: "; cin >> str1; // Không nhận khoảng trắng
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Nhập str2: "; fgets(str2, 100, stdin); // Ký tự \n sẽ được đưa vào cuối chuỗi

    cout << "Nhập str3: "; cin.get(str3, 100); // Ký tự \n sẽ không mất đi trên Stream
    cin.ignore();

    cout << "Nhập str4: "; cin.getline(str4, 100);

    cout << "Nhập str5: "; getline(cin, str5);

    cout << "\nXuất str1: " << str1 << endl;
    cout << "Xuất str2: "; puts(str2);
    cout << "Xuất str3: "; puts(str3);
    cout << "Xuất str4: " << str4 << endl;
    cout << "Xuất str5: " << str5 << endl;
    return 0;
}
```

## Kết quả thực thi:

Nhap str1: chuoi thu nhat  
Nhap str2: chuoi thu hai  
Nhap str3: chuoi thu ba  
Nhap str4: chuoi thu tu  
Nhap str5: chuoi thu nam

Xuat str1: chuoi  
Xuat str2: chuoi thu hai

Xuat str3: chuoi thu ba  
Xuat str4: chuoi thu tu  
Xuat str5: chuoi thu nam



## 3.8.4 Xóa bộ nhớ đệm

- Xét đoạn mã sau:

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    int n;
    string s;
    cout << "Nhập n: ";
    cin >> n;

    cout << "Nhập chuỗi s: ";
    getline(cin, s);
    cout << endl;
    cout << "n = " << n << endl;
    cout << "s = " << s << " !" << endl;
    return 0;
}
```

### Kết quả thực thi:

```
Nhập n: 123
Nhập chuỗi s:
n = 123
s = !
```



## 3.8.4 Xóa bộ nhớ đệm

- Cách giải quyết: kết hợp các lệnh sau: `cin.ignore(); getc(stdin); getchar(); cin.get();`

```
cout << "Nhập n: ";
cin >> n;
```

```
cin.ignore(); // getc(stdin); getchar(); cin.get();
```

```
cout << "Nhập chuỗi s: ";
getline(cin, s);
```

**Kết quả thực thi:**

```
Nhập n: 123
Nhập chuỗi s: Vi du
n = 123
s = Vi du !
```



### 3.8.4 Xóa bộ nhớ đệm

- Mọi ký tự được gõ vào từ bàn phím đều được đưa vào bộ nhớ đệm stdin hoặc stream trước khi được gán vào biến.
- Một số hàm nhập (cin, cin.get, ...) không xử lý ký tự kết thúc việc nhập vào '\n' → ký tự Enter vẫn còn trong bộ nhớ đệm. Đến khi nhập chuỗi, các hàm nhập chuỗi nhận được ký tự Enter trên bộ nhớ đệm thì dừng nhập. Điều này khiến ta sẽ không nhập được giá trị cho chuỗi mong muốn.
- Vì vậy trước khi nhập một chuỗi trong bộ nhớ đệm hoặc stream, ta cần có thao tác xóa bộ nhớ đệm để đảm bảo việc nhập như mong muốn.



## 3.8.4 Xóa bộ nhớ đệm

- Các hàm sau dùng để xóa bộ nhớ đệm:

Hàm	Thư viện	Ý nghĩa
<code>fflush(stdin)</code>	<code>&lt;stdio.h&gt;</code>	Dùng để xóa bộ nhớ đệm cho stream nhập từ bàn phím.
<code>cin.ignore(streamsize n = 1, int delim = EOF)</code> • Ví dụ: <code>cin.ignore(numeric_limits&lt;stream size&gt;::max(), '\n');</code>	<code>&lt;iostream&gt;</code>	Tách và bỏ qua n các ký tự trong bộ nhớ đệm stdin. Bỏ qua hay loại bỏ một số lượng ký tự n trong bộ đệm stdin, số lượng ký tự còn lại (nếu có) vẫn ở trên stdin hoặc bỏ qua đến khi gặp ký tự <i>delim</i> .
<code>std::cin.ignore()</code>	<code>&lt;iostream&gt;</code>	Mặc định là bỏ 1 ký tự trong bộ đệm.



# Ví dụ 1:

```
// cin with number and string
#include <iostream> // std::cout,
std::cin, std::getline
#include <string> // std::string
#include <stdio.h> // fflush(stdin)
int main () {
    std::string name;
    int age;
    std::cout << "Age: ";
    std::cin >> age;

    std::cin.ignore(6, '\n');
    std::cout << "Name: ";
    std::getline(std::cin, name);

    std::cout << std::endl <<
    std::endl << "Output: " << name
    << ", " << age;
    return 0;
}
```

Chạy lần 1:

Age: 35 123

// std::cin.ignore(6, '\n'): gặp enter nên kết thực việc nhập, dãy "123" gồm 4 ký tự bị bỏ qua, stdin rỗng

Name: ngoc diem

Output: ngoc diem, 35

-----

Chạy lần 2:

Age: 35 123456789

// std::cin.ignore(6, '\n'): gặp enter nên kết thực việc nhập, dãy "12345" gồm 6 ký tự bị bỏ qua, stdin còn lại là dãy "6789"

Name:

Output: 6789, 35



## Ví dụ 2:

```
#include <iostream> // cin, cout
Using namespace std;
int main () {
    char first, last;
    cout << "surname: ";
    first = cin.get(); // get one
    character
    cin.ignore(6, '\n'); // ignore until
    enter
    cout << "lastname: ";
    cin >> last; // get one character
    cout << endl << "Your initials: " <<
    first << last << '\n';

    return 0;
}
```

Chạy lần 1:

surname: d123456789

// std::cin.ignore(6, '\n'): gấp enter nên kết thực việc nhập,  
dãy “ 123456” gồm 6 ký tự bị bỏ qua, stdin còn lại là dãy  
“789”

lastname:

Your initials: d7

-----

Chạy lần 2:

surname: d123

// std::cin.ignore(6, '\n'): gấp enter nên kết thực việc nhập,  
dãy “ 123” gồm 3 ký tự bị bỏ qua, stdin rỗng

lastname: 456

//gấp enter nên kết thực việc nhập, stdin còn lại là dãy “56”

Your initials: d4



# Ví dụ

```
4  using namespace std;
5  int main(){
6      string HoTen="Nguyen Mai"; char GioiTinh='F';
7      float DiemToan=9.5; string Lop="Nhap mon lap trinh";
8
9      getline(cin, HoTen);
10     cin >> GioiTinh; cin.ignore(256, '\n');
11     cin >> DiemToan; getline(cin, Lop);
12
13     cout << endl << endl;
14     cout << HoTen << endl;
15     cout << GioiTinh << endl;
16     cout << DiemToan << endl;
17     cout << Lop << endl;
```

```
D:\vd\bin\Debug\vd.exe
Mai Nguyen
F123Nhapmon
9.75

Mai Nguyen
F
9.75
```



## 3.8 Các ví dụ



# Ví dụ 1

- Viết chương trình Nhập, xuất thông tin của 1 sinh viên. Biết thông tin của sinh viên bao gồm: { Họ tên, Giới tính, Điểm toán, Điểm tin, Điểm trung bình, Xếp loại, Kết quả}

Biến	Kiểu dữ liệu	Ví dụ
Họ tên sinh viên	(chuỗi)	"Nguyen Xuan Minh"
Giới tính	(char)	M (Male)/ F (Female)
Điểm Toán	(số nguyên)	8
Điểm Tin	(số nguyên)	9
Điểm Trung bình	float	8.5
Xếp loại	Chuỗi	Giỏi
Kết quả	bool	1 (Đậu)/ 0 (Rớt)



## Ví dụ 2

- Nhập năm sinh của một người và tính tuổi của người đó.

```
#include <iostream>
int main() {
    int namsinh = 0;
    std::cout << "Vui long nhap nam sinh: ";
    std::cin >> namsinh;
    std::cout << "Ban " << 2016-namsinh << " tuoi ." << endl;
    return 0;
}
```



## Ví dụ 3

- Nhập 2 số a và b. Tính tổng, hiệu, tích và thương của hai số đó.

```
#include <iostream>
int main() {
    int a, b;
    std::cout << "Nhập a = ";
    std::cin >> a;
    std::cout << "Nhập b = ";
    std::cin >> b;
    std::cout << "a + b = " << a+b << "\n";
    std::cout << "a - b = " << a-b << "\n";
    std::cout << "a * b = " << (long long)a*b << "\n";
    std::cout << "a / b = " << (double)a/b << "\n";
    return 0;
}
```



## Ví dụ 4

Bài 3. Nhập tên sản phẩm, số lượng và đơn giá. Tính tiền và thuế giá trị gia tăng phải trả, biết:

- a. tiền = số lượng \* đơn giá
- b. thuế giá trị gia tăng = 10% tiền

```
#include <iostream>
int main() {
    int so_luong = 0, don_gia = 0;
    std::cout << "Vui lòng nhập số lượng: ";
    std::cin >> so_luong;
    std::cout << "Vui lòng nhập đơn giá: ";
    std::cin >> don_gia;
    std::cout << "Tien: " << so_luong * don_gia << "\n";
    std::cout << "VAT: " << so_luong * don_gia * 0.1 << "\n";
    return 0;
}
```



## Ví dụ 5

Bài 4. Nhập bán kính của đường tròn. Tính chu vi và diện tích của hình tròn đó (độ chính xác sau dấu thập phân là 3).

```
#include <iostream>
#define PI 3.14
int main() {
    float r = 0;
    std::cout << "Nhập bán kính đường tròn: ";
    std::cin >> r;
    std::cout << "Chu vi: " << 2 * PI * r << "\n";
    std::cout << "Diện tích: " << PI * r * r << "\n";
    return 0;
}
```



# Bài tập

- Câu 1: Cho số xe (gồm 4 chữ số) của bạn. Cho biết số xe của bạn được mấy nút?
- Câu 2: Cho 1 ký tự chữ thường. In ra ký tự chữ hoa tương ứng.
- Câu 3: Cho 3 số nguyên. Cho biết số lớn nhất và nhỏ nhất?
- Câu 4: Viết chương trình cho 2 giờ (giờ, phút, giây) và thực hiện cộng, trừ 2 giờ này.
- Câu 5: Tính tổng các bội số của 3 và 5 nhỏ hơn 1000.  
Ví dụ: Tổng các bội số của 3 và 5 nhỏ hơn 10 là 23 (các bội số: 3, 5, 6, 9 → Tổng: 23)



# Chúc các em học tốt !

