

# Bài 10: NHẬP, XUẤT VÀ ĐỊNH DẠNG DỮ LIỆU TRONG C++ (INPUT AND OUTPUT)

Xem bài học trên website để ủng hộ Kteam: [Nhập, Xuất và Định dạng dữ liệu trong C++ \(Input and Output\)](#)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage [How Kteam](#) nhé!

## Dẫn nhập

Ở bài học trước, bạn đã nắm được [KIỂU LUÂN LÝ & CƠ BẢN VỀ CÂU ĐIỀU KIỆN IF TRONG C++ \(Boolean and If statements basic\)](#). Trong mỗi bài học trước, đều có những ví dụ liên quan đến việc xuất một thông tin nào đó ra màn hình console, nhưng có thể mình chưa nói kỹ về phần này.

Hôm nay, mình sẽ giải thích chi tiết về **Nhập, Xuất và Định dạng dữ liệu trong C++ (Input and Output)**.

---

## Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- [BIẾN TRONG C++ \(Variables\)](#)
- [CÁC KIỂU DỮ LIỆU CƠ BẢN TRONG C++ \(Integer, Floating point, Character, Boolean\)](#)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Xuất dữ liệu với `std::cout` trong C++
- Xuất dữ liệu với `std::cin` trong C++
- Định dạng dữ liệu nhập xuất trong C++

## Xuất dữ liệu với `std::cout` trong C++

Đối tượng **`std::cout`** là một đối tượng **được định nghĩa trong iostream library thuộc namespace std**, dùng để hiển thị một thông tin nào đó lên thiết bị xuất chuẩn (mặc định là màn hình). Toán tử **`<<` (insertion operator)** được **dùng chung với `std::cout`**, cho biết **hướng đi của data từ r-value đến màn hình console**.

Trong mỗi bài học trước, đều có những ví dụ liên quan đến việc sử dụng đối tượng **`std::cout`** để xuất một thông tin nào đó ra màn hình console. Một ví dụ kinh điển về chương trình mà bất cứ một developer nào cũng từng viết mỗi khi học một ngôn ngữ mới:

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

int main()
{
    cout << "Hello HowKteam.com"; // in lên màn hình dòng chữ "Hello
    HowKteam.com"
    return 0;
}
```

Bạn có thể sử dụng toán tử **`<<` (insertion operator)** nhiều lần để in nhiều thông tin trên cùng một dòng. **Ví dụ:**

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

int main()
```

```
{  
    cout << "Hello HowKteam.com" << endl; // sử dụng 2 lần toán tử <<  
  
    int n1{ 60 };    // n1 = 60  
    int n2{ 9 };    // n2 = 9  
  
    // in chuỗi "Sum: 60 + 9 = 69" và xuống dòng  
    cout << "Sum: " << n1 << " + " << n2 << n1 + n2 << "\n";  
    return 0;  
}
```

## Newline '\n' và std::endl

Đến đây, có lẽ sẽ có một số bạn vẫn thắc mắc về **sự khác nhau giữa đối tượng std::endl và escape sequence '\n'**.

Nếu bạn viết một chương trình như bên dưới và sử dụng cả 2 cách, bạn sẽ có được kết quả như nhau:

```
std::cout << "HowKteam.com" << std::endl;  
std::cout << "Free education\n";
```

Tuy nhiên, 2 cách này có thực sự giống nhau? **Câu trả lời là không, bản chất của std::endl** được thể hiện ở 2 câu lệnh bên dưới:

```
std::cout << "HowKteam.com" << std::endl;  
  
// Tương đương với:  
  
std::cout << "HowKteam.com\n" << std::flush;
```

Trong C++, **output stream thường dùng buffer**, nghĩa là **output data** sẽ **được lưu vào một vùng nhớ đệm**, và **output data** sẽ **được gửi đến output device** vào thời điểm thích hợp (vì lý do hiệu suất).

Với **std::endl** sẽ **xóa output buffer** mỗi khi nó được gọi, trong khi **'\n'** thì không.

Vậy, khi nào nên sử dụng **std::endl** và **'\n'**:

- **Nên sử dụng std::endl** khi bạn cần **đảm bảo output** của bạn có **ngay lập tức** (Vd: khi viết một record vào một file, hoặc khi update một thành phần tiến trình). Nhưng nên hạn chế sử dụng **std::endl** khi làm việc với file I/O để tránh việc phải flush buffer liên tục dẫn đến việc phải truy cập các file I/O thường xuyên (giảm hiệu suất).
- Ngoài ra, những trường hợp khác nên sử dụng **'\n'**.

## Nhập dữ liệu với std::cin trong C++

Đối tượng **std::cin** là một đối tượng **được định nghĩa trong iostream library thuộc namespace std**, dùng để **đọc một thông tin nào đó từ thiết bị nhập chuẩn** (mặc định là bàn phím), **sau đó lưu thông tin đó vào một biến**. Toán tử **>> (extraction operator)** được dùng chung với **std::cin**, cho biết **hướng đi của data từ màn hình console vào một biến**.

Bên dưới là một chương trình yêu cầu người dùng nhập một số, sau đó xuất số vừa nhập ra màn hình:

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

int main()
{
    int n{ 0 };

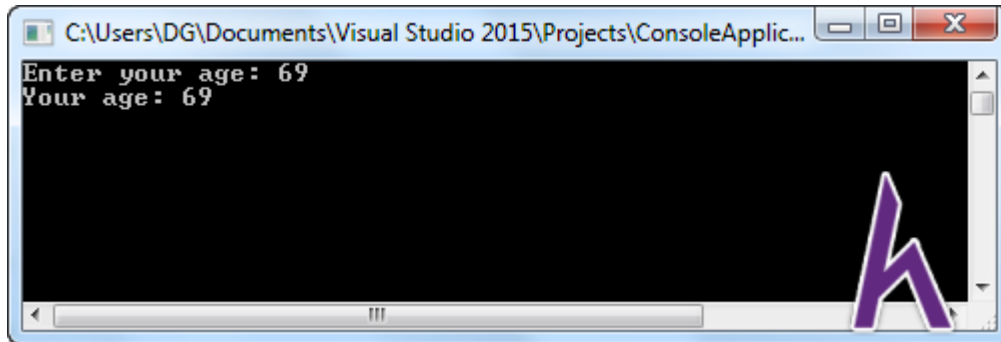
    // thông báo yêu cầu user nhập tuổi
    cout << "Enter your age: ";

    // đọc giá trị từ console và lưu vào biến n
    cin >> n;

    // in giá trị biến n (tuổi) lên màn hình
```

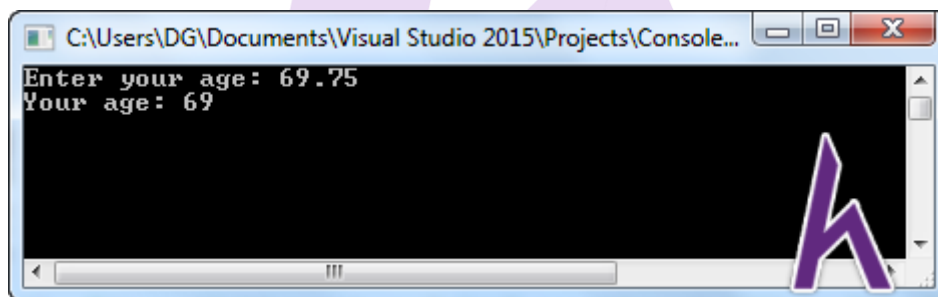
```
cout << "Your age: " << n << endl;  
return 0;  
}
```

### Outputs:



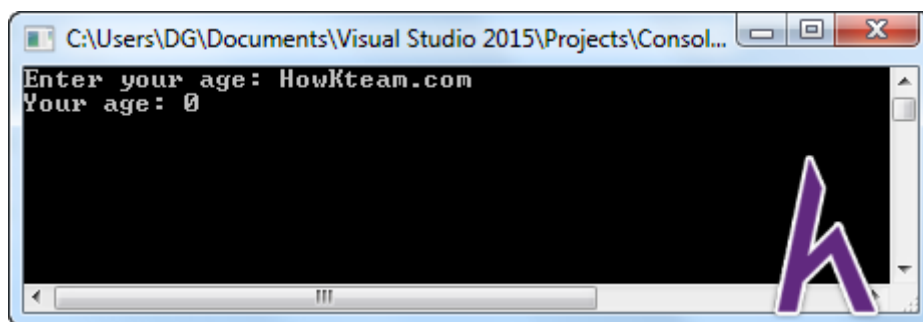
Ở chương trình trên, **nếu bạn nhập vào một số thực**, khi số đó được lưu vào biến **n** thì **C++ sẽ ép kiểu ngầm định số thực vừa nhập về số nguyên**, tức là **phần thập phân sẽ bị mất đi**.

### Ví dụ:



Nếu bạn đang cố gắng **nhập một giá trị bất kỳ không phải là số**, hoặc một **số nằm ngoài phạm vi kiểu dữ liệu của biến đó**, thì **giá trị đó sẽ không được gán cho biến**. Lúc này, giá trị biến sẽ không thay đổi.

### Ví dụ:



## Nhập nhiều giá trị liên tiếp trong một câu lệnh

Giống như **std::cout**, bạn cũng có thể nhập giá trị cho nhiều biến bằng cách sử dụng nhiều toán tử **>>** (**extraction operator**) trong một câu lệnh. Đối tượng **std::cin** sẽ lấy mỗi giá trị theo ký tự khoảng trắng, hoặc ký tự xuống dòng từ trái qua phải và từ trên xuống dưới.

### Ví dụ:

```
#include <iostream>
#include <iomanip>      // for std::setprecision()
using namespace std;

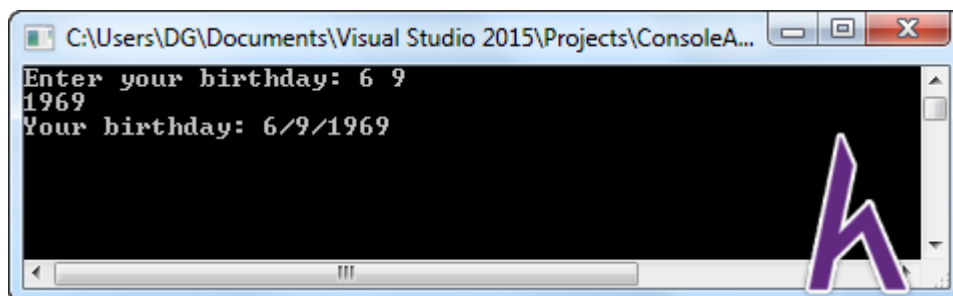
int main()
{
    int    nDay{ 0 };
    int    nMonth{ 0 };
    int    nYear{ 0 };

    // thông báo yêu cầu user nhập ngày, tháng, năm sinh
    cout << "Enter your birthday: ";

    // đọc giá trị từ console và lưu vào biến nDay, nMonth, nYear
    cin >> nDay >> nMonth >> nYear;

    // in 3 giá trị vừa nhập lên màn hình
    cout << "Your birthday: " << nDay << "/" << nMonth << "/" << nYear
    << endl;
    return 0;
}
```

## Outputs:



Trong chương trình trên, 6 và 9 cách nhau bởi ký tự khoảng trắng, 1969 nằm ở một dòng mới. Đối tượng `std::cin` đã lưu được 3 giá trị vào mỗi biến tương ứng.

## Định dạng dữ liệu nhập xuất trong C++

Trong C++, bạn có thể **định dạng dữ liệu nhập xuất cho thiết bị nhập xuất chuẩn** (bàn phím, màn hình console), hoặc có thể **định dạng dữ liệu nhập xuất cho file văn bản**.

Để định dạng dữ liệu, bạn cần **thêm chỉ thị `#include <iomanip>`** vào đầu chương trình. **Thư viện này chứa các toán tử định dạng (manipulator)**.

**Ví dụ: `std::endl` cũng là một manipulator thuộc `<iostream>` library.** Bên dưới là những manipulator khá thông dụng trong C++:

- Toán tử **`std::setw(n)`**: xác định độ rộng dành cho của dữ liệu xuất. Khi sử dụng **`std::setw(n)`**, các khoảng trắng sẽ được thêm vào bên **trái hoặc bên phải dữ liệu xuất** ( để tổng số ký tự là `n`). Dữ liệu khi in ra sẽ được **canh trái hoặc canh phải**.
- Toán tử **`std::left`** và **`std::right`** dùng chung với **`std::setw(n)`** để canh lề **trái hoặc lề phải**.
- Toán tử **`std::setfill(ch)`** dùng chung với **`std::setw(n)`** để **quy định ký tự `ch` được thêm vào thay vì dùng khoảng trắng mặc định**. Ví dụ: nếu dùng **`std::setfill('-')`** thì dấu '-' sẽ được thay cho khoảng trắng.

**Ví dụ:**

```
#include <iostream>
#include <iomanip>      // for std::setw(n), std::setfill(ch), std::left, std::right
using namespace std;

int main()
{
    cout << "Kteam Solutions and Entertainment" << endl;
    cout << "HowKteam.com" << endl << endl;

    cout << setw(5) << left << "ID";          // độ rộng 5 ký tự, canh trái ID
    cout << setw(30) << left << "Name";       // độ rộng 30 ký tự, canh trái
Name
    cout << setw(20) << right << "Address" << endl; // độ rộng 20 ký tự,
canh phải Address

    cout << setfill('-');                    // set fill bằng ký tự '-' thay vì ' '
    cout << setw(55) << "-" << endl; // fill 55 ký tự '-'

    // reset fill bằng ký tự ' '
    cout << setfill(' ');

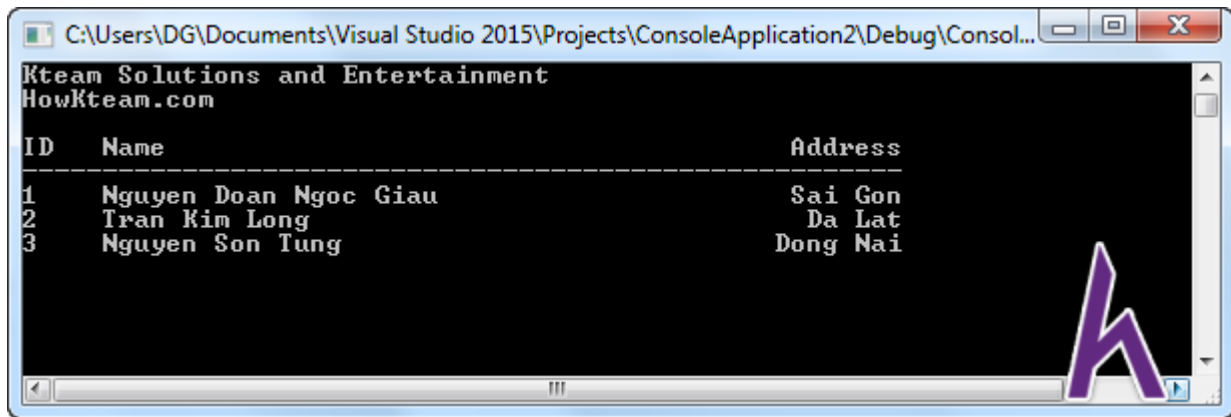
    // in thông tin theo format như trên
    cout << setw(5) << left << 1;
    cout << setw(30) << left << "Nguyen Doan Ngoc Giau";
    cout << setw(20) << right << "Sai Gon" << endl;

    cout << setw(5) << left << 2;
    cout << setw(30) << left << "Tran Kim Long";
    cout << setw(20) << right << "Da Lat" << endl;

    cout << setw(5) << left << 3;
    cout << setw(30) << left << "Nguyen Son Tung";
    cout << setw(20) << right << "Dong Nai" << endl;
    return 0;
}
```

**Outputs:**





- Các toán tử **std::dec** (thập phân), **std::oct** (bát phân), **std::hex** (thập lục phân) quy định số nguyên khi nhập xuất theo dạng thập phân, bát phân, hay thập lục phân.

### Ví dụ:

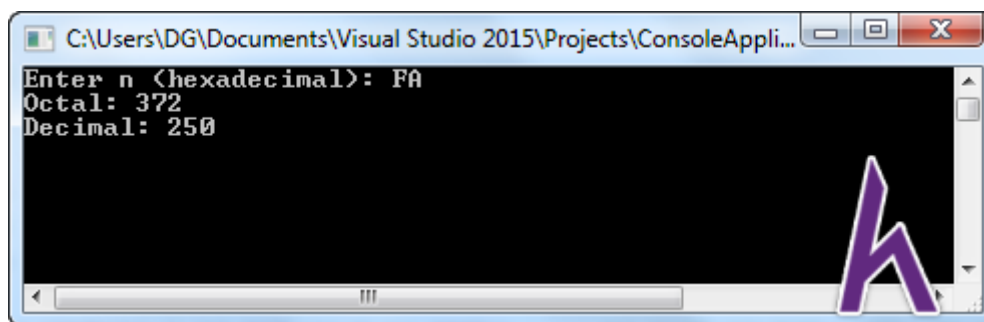
```
#include <iostream>
#include <iomanip>      // for std::hex, std::oct, std::dec
using namespace std;

int main()
{
    int n;
    cout << "Enter n (hexadecimal): ";
    cin >> hex >> n; // nhập số thập lục phân (hệ 16)

    cout << "Octal: " << oct << n << endl;    // xuất số bát phân (hệ 8)
    cout << "Decimal: " << dec << n << endl; // xuất số thập phân (hệ 10)

    return 0;
}
```

### Outputs:



- Toán tử **`std::setprecision(n)`** quy định độ chính xác khi in số thực, n là tổng các chữ số khi in. Toán tử này đã được hướng dẫn kỹ trong bài [SỐ TƯ NHIÊN VÀ SỐ CHẤM ĐÔNG TRONG C++ \(Integer, Floating point\)](#).

---

## Kết luận

Qua bài học này, bạn đã nắm được các thao tác Nhập, Xuất và Định dạng dữ liệu trong C++ (Input and Output), và đã biết được những kinh nghiệm cũng như kỹ thuật liên quan đến nhập xuất trong C++.

Ở bài tiếp theo, bạn sẽ được học một khái niệm mới có liên quan đến biến (variables) và rất hay gặp trong lập trình, đó là: [HẰNG SỐ TRONG C++ \(Constants\)](#)

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên **"Luyện tập – Thử thách – Không ngại khó"**.