C++ References (Tham chiếu)

Chương 6: Bí mật về những "Biệt danh"

Slide Learning CPP

Ngày 14 tháng 1 năm 2026

Khởi động: Bức tranh toàn cảnh

Chào mừng bạn đến với thế giới của References!

Ví dụ: Nguyễn Văn An và Tí

Hãy tưởng tượng bạn có một người bạn tên là **"Nguyễn Văn An"**. Ở nhà, bố mẹ gọi bạn ấy là **"Tí"**.

- Dù gọi là "An"hay "Tí", thì **người đó vẫn là một**.
- Nếu "Tí"bị dính mực, áo của "An"cũng bị dính mực.

Trong C++, **Biến chính** là tên khai sinh, còn **Reference** chính là biệt danh.

Lộ trình khám phá

Chúng ta sẽ đi qua 3 phần chính:

1. **Chương 1: Tạo ra "Biệt danh"(Creating References)**
2. **Chương 2: Phép thuật "Tuy hai mà một"(How it works)**
3. **Chương 3: Phân biệt "Biệt danh"và "Địa chỉ nhà"(Reference vs. Memory Address)**

Chương 1: Tạo ra "Biệt danh"

Ẩn dụ: Nhãn dán trên chiếc hộp

1. Bạn có một chiếc hộp đựng **Pizza** dán nhãn `mon_an`.
2. Bạn dán thêm nhãn thứ hai lên **chính hộp đó** tên là `mon_phu`.

→ Lấy đồ từ hộp nào cũng ra cùng một chiếc Pizza.

Cú pháp C++: Dùng ký hiệu **&** (dấu và).

```
kieu_du_lieu &ten_biet_danh = ten_bien_goc;
```

Code ví dụ: Tạo Reference

Hãy xem cách máy tính xử lý "biệt danh":

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

int main() {
  // 1. Tao bien goc (Cai nhan thu nhat)
  string mon_an = "Pizza";

  // 2. Tao bien tham chieu (Cai nhan thu hai)
  // Chu y dau & o day nhe!
  string &bua_trua = mon_an;

  cout << "Ten bien goc: " << mon_an << "\n";
  cout << "Ten biet danh: " << bua_trua << "\n";

  return 0;
}
```

Kết quả: Cả hai đều in ra "Pizza".

Góc kiểm tra nhanh

Câu đố

Nếu trong đoạn code trước, mình thay đổi giá trị của `mon_an` thành **"Banh Mi"**.
Lúc đó in `bua_trua` ra, nó sẽ là "Pizza"(cũ) hay "Banh Mi"(mới)?

Góc kiểm tra nhanh

Câu đố

Nếu trong đoạn code trước, mình thay đổi giá trị của `mon_an` thành **"Banh Mi"**.
Lúc đó in `bua_trua` ra, nó sẽ là "Pizza"(cũ) hay "Banh Mi"(mới)?

Đáp án: Banh Mi!

Chính xác! Vì chúng cùng dán trên một chiếc hộp, nên thay ruột hộp thì nhãn nào cũng "đọc"ra món mới cả.

Chương 2: Phép thuật "Tuy hai mà một"

Ẩn dụ: Chiếc TV và hai cái điều khiển

Hãy tưởng tượng biến trong C++ như một chiếc TV.

- **Biến gốc (`mon_an`):** Điều khiển màu Đỏ.
- **Tham chiếu (`bua_trua`):** Điều khiển màu Xanh.

Cả hai cùng kết nối tới **MỘT** chiếc TV.

Nếu bố dùng điều khiển Xanh bật "Bún Chả", bạn nhìn màn hình cũng sẽ thấy "Bún Chả".

Code thử nghiệm: Thay đổi giá trị

Sức mạnh của Reference: Can thiệp trực tiếp vào biến gốc.

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

int main() {
  string mon_an = "Pizza";       // Man hinh dang chieu Pizza
  string &bua_trua = mon_an;     // Ket noi them dieu khien 2

  // Thay doi gia tri thong qua "Biet danh"
  bua_trua = "Bun Cha";

  // Kiem tra lai bien goc
  cout << "Mon an bay gio la: " << mon_an << "\n";

  return 0;
}
```

Kết quả

Mon an bay gio la: Bun Cha

Trạm dừng chân suy ngẫm

Câu hỏi quan trọng

Trong ví dụ trên, biến `mon_an` và biến `bua_trua` có phải là 2 biến khác nhau chiếm 2 chỗ trong bộ nhớ không? Hay chỉ 1 chỗ?

Trạm dừng chân suy ngẫm

Câu hỏi quan trọng

Trong ví dụ trên, biến `mon_an` và biến `bua_trua` có phải là 2 biến khác nhau chiếm 2 chỗ trong bộ nhớ không? Hay chỉ 1 chỗ?

Đáp án: Chỉ 1 chỗ!

- Chúng dùng chung vùng nhớ.
- **Lợi ích:** Giúp chương trình chạy nhanh hơn vì không phải copy dữ liệu.

Chương 3: Phân biệt "Biệt danh"và "Địa chỉ"

Ký hiệu **&** đóng 2 vai khác nhau tùy ngữ cảnh:

Vai 1: Reference

- Xuất hiện khi **khai báo biến mới**.
- Cú pháp: `string &x = ...`
- Ý nghĩa: Tạo biệt danh.
- *Ví dụ: Tấm biển tên trước cổng.*

Vai 2: Address Operator

- Đứng trước biến **đã tồn tại**.
- Cú pháp: `&x`
- Ý nghĩa: Lấy địa chỉ bộ nhớ.
- *Ví dụ: Số nhà (123 đường ABC).*

Chứng minh: Hai tên gọi, Một địa chỉ

```cpp
int main() {
  string mon_an = "Pizza";
  string &bua_trua = mon_an; // Dung & de tao biet danh

  // 1. In ra gia tri
  cout << "Gia tri: " << mon_an << "\n";

  // 2. In ra dia chi (Dung & truoc ten bien)
  cout << "Dia chi mon_an:   " << &mon_an << "\n";
  cout << "Dia chi bua_trua: " << &bua_trua << "\n";

  return 0;
}
```

Kết quả (Minh họa)

Gia tri: Pizza
Dia chi mon_an: 0x6dfed4
Dia chi bua_trua: 0x6dfed4

Địa chỉ giống hệt nhau → Cùng một vị trí bộ nhớ.

Tổng kết hành trình

1. **Reference là gì?** Là "biệt danh"cho một biến đã có.
2. **Đặc điểm:** Không tạo bản copy, dùng chung vùng nhớ. Thay đổi một, ảnh hưởng tất cả.
3. **Phân biệt ký hiệu &:**
   - `int &x`: Khai báo tham chiếu.
   - `&x`: Lấy địa chỉ bộ nhớ.

Thử thách Code nhỏ

Đề bài

Viết đoạn code ngắn:

1. Tạo biến `diem_so` bằng 10.
2. Tạo tham chiếu `ket_qua` trỏ tới `diem_so`.
3. Thay đổi `ket_qua` thành 100.
4. In ra `diem_so`.

Thử thách Code nhỏ

Đề bài

Viết đoạn code ngắn:

1. Tạo biến `diem_so` bằng 10.
2. Tạo tham chiếu `ket_qua` trỏ tới `diem_so`.
3. Thay đổi `ket_qua` thành 100.
4. In ra `diem_so`.

Đáp án (Chuẩn ngữ pháp)

```
1 int diem_so = 10;
2 int &ket_qua = diem_so;
3 ket_qua = 100;
4
5 // Ket qua in ra se la 100
6 cout << diem_so;
```

Bước tiếp theo: Trùm cuối xuất hiện

Bạn đã làm chủ được References (Biệt danh).

Next Step: POINTERS (CON TRỎ)

Nếu Reference là "Biệt danh", thì Pointers là làm việc trực tiếp với "Địa chỉ nhà"(`0x...`) mà chúng ta vừa thấy.

Bạn đã sẵn sàng để gặp "Trùm cuối"chưa?