NGÔN NGỮ LẬP TRÌNH

Bài 1: Giới thiệu ngôn ngữ lập trình C++

Giảng viên: Lý Anh Tuấn

Email: tuanla@tlu.edu.vn

Nội dung

1. Giới thiệu C++
   - Nguồn gốc, Lập trình hướng đối tượng, Các thuật ngữ
2. Biến, Biểu thức, Lệnh gán
3. Vào ra dữ liệu
4. Phong cách lập trình
5. Thư viện và Không gian tên

Giới thiệu C++

- Nguồn gốc C++
  - Ngôn ngữ bậc thấp: *Mã máy*, *Assembly*
  - Ngôn ngữ bậc cao: *C*, *C++*, *FORTRAN*, *COBOL*
  - Lập trình hướng đối tượng trong C++
- Thuật ngữ C++
  - Chương trình (*Program*), Hàm (*Function*), Thư viện (*Library*)
  - Vào ra cơ bản (IO) với *cin*/*cout*

Chương trình C++ ví dụ (1/2)

Display 1.1 A Sample C++ Program

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

3   int main( )
4   {
5       int numberOfLanguages;

6       cout << "Hello reader.\n"
7            << "Welcome to C++.\n";

8       cout << "How many programming languages have you used? ";
9       cin >> numberOfLanguages;

10      if (numberOfLanguages < 1)
11          cout << "Read the preface. You may prefer\n"
12               << "a more elementary book by the same author.\n";
13      else
14          cout << "Enjoy the book.\n";

15      return 0;
16  }
```

Chương trình C++ ví dụ (2/2)

SAMPLE DIALOGUE 1

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 0
Read the preface. You may prefer
a more elementary book by the same author.
```

User types in 0 on the keyboard.

SAMPLE DIALOGUE 2

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 1
Enjoy the book
```

User types in 1 on the keyboard.

Biến trong C++

- Định danh trong C++
  - Không trùng với từ khóa hoặc từ dành riêng
  - Phân biệt chữ hoa, chữ thường
  - Nên là tên có nghĩa
- Biến trong C++
  - Là một vùng bộ nhớ để lưu trữ dữ liệu cho một chương trình
  - Phải được khai báo trước khi sử dụng trong chương trình

Các kiểu dữ liệu (1/2)

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED	SIZE RANGE	PRECISION
`short` (also called `short int`)	2 bytes	–32,767 to 32,767	Not applicable
`int`	4 bytes	–2,147,483,647 to 2,147,483,647	Not applicable
`long` (also called `long int`)	4 bytes	–2,147,483,647 to 2,147,483,647	Not applicable
`float`	4 bytes	approximately 10^{-38} to 10^{38}	7 digits
`double`	8 bytes	approximately 10^{-308} to 10^{308}	15 digits

Các kiểu dữ liệu (2/2)

`long double`	10 bytes	approximately 10^{-4932} to 10^{4932}	19 digits
`char`	1 byte	All ASCII characters (Can also be used as an integer type, although we do not recommend doing so.)	Not applicable
`bool`	1 byte	`true`, `false`	Not applicable

The values listed here are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. *Precision* refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types `float`, `double`, and `long double` are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.

Gán dữ liệu cho biến

- Khởi tạo dữ liệu trong câu lệnh khai báo
  - Nếu chưa được khởi tạo, biến sẽ có giá trị "undefined"!
    - *int myVar = 0;*
- Gán dữ liệu trong khi chạy chương trình
  - Vế trái & Vế phải
    - **Vế trái** phải là biến
    - **Vế phải** có thể là bất kỳ biểu thức nào
    - Ví dụ: *distance = rate * time;*
      - **Vế trái** là *distance*
      - **Vế phải** là *rate * time*

Gán dữ liệu: Ký hiệu viết tắt (1/2)

EXAMPLE	EQUIVALENT TO
`count += 2;`	`count = count + 2;`
`total -= discount;`	`total = total - discount;`
`bonus *= 2;`	`bonus = bonus * 2;`
`time /= rushFactor;`	`time = time/rushFactor;`
`change %= 100;`	`change = change % 100;`
`amount *= cnt1 + cnt2;`	`amount = amount * (cnt1 + cnt2);`

Những quy tắc gán dữ liệu

- Tính tương thích của gán dữ liệu
  - Khác kiểu dữ liệu:
    - Quy tắc: không gán giá trị thuộc một kiểu dữ liệu cho biến thuộc một kiểu dữ liệu khác
  - intVar = 2.99; // 2 được gán cho intVar!
    - Chỉ phần nguyên là hợp kiểu, vì thế chỉ nó được gán cho biến intVar
    - Được gọi là "*ép kiểu ngầm*" hoặc "*chuyển đổi dữ liệu tự động*"
  - 2, 5.75, "Z", "Hello World" được xem là các *hằng literal*, không thể thay đổi giá trị khi chạy chương trình

Chuỗi escape

- Cú pháp: \<ký_tự>
- Báo với trình biên dịch đây là một ký tự đặc biệt

Display 1.3 Some Escape Sequences

SEQUENCE	MEANING
\n	New line
\r	Carriage return (Positions the cursor at the start of the current line. You are not likely to use this very much.)
\t	(Horizontal) Tab (Advances the cursor to the next tab stop.)
\a	Alert (Sounds the alert noise, typically a bell.)
\\	Backslash (Allows you to place a backslash in a quoted expression.)

Chuỗi escape

\'	Single quote (Mostly used to place a single quote inside single quotes.)
\"	Double quote (Mostly used to place a double quote inside a quoted string.)
The following are not as commonly used, but we include them for completeness:	
\v	Vertical tab
\b	Backspace
\f	Form feed
\?	Question mark

Hằng

- Đặt tên cho hằng
  - Hằng literal ít mang ý nghĩa. VD: số 24 không cho biết nó diễn đạt thông tin gì
- Hằng được đặt tên cung cấp ý nghĩa nó muốn diễn đạt
  - VD: *const int NUMBER_OF_STUDENTS = 24;*
    - Là hằng được khai báo hay hằng được đặt tên
    - Có thể sử dụng tên của hằng bất cứ chỗ nào trong chương trình

Hằng được đặt tên

Display 1.4 **Named Constant**

```
1   #include <iostream>
2   using namespace std;
3
4   int main( )
5   {
6       const double RATE = 6.9;
7       double deposit;

8       cout << "Enter the amount of your deposit $";
9       cin >> deposit;
10      double newBalance;
11      newBalance = deposit + deposit*(RATE/100);
12      cout << "In one year, that deposit will grow to\n"
13           << "$" << newBalance << " an amount worth waiting for.\n";

14      return 0;
15  }
```

SAMPLE DIALOGUE

Enter the amount of your deposit $**100**
In one year, that deposit will grow to
$106.9 an amount worth waiting for.

Độ chính xác phép toán

- Biểu thức trong C++ có thể được tính toán không như bạn mong đợi
- Toán hạng bậc cao nhất sẽ quyết định kiểu độ chính xác phép toán được thực hiện
- Ví dụ:
  - 17 / 5 trả về giá trị 3
    - Cả hai toán hạng là số nguyên
    - Phép chia số nguyên được thực hiện!
  - 17.0 / 5 trả về giá trị 3.4
    - Toán hạng bậc cao nhất có kiểu thực
    - Phép chia độ chính xác số thực được thực hiện!
  - int var1 = 1, var2 = 2;
    - var1/var2 = ?

Độ chính xác phép toán

- Các phép toán được thực hiện từng bước một
  - 1 / 2 / 3.0 / 4 : Thực hiện 3 phép chia riêng rẽ
    - đầu tiên 1 / 2 bằng 0
    - tiếp đó 0 / 3.0 bằng 0.0
    - tiếp đó 0.0 / 4 bằng 0.0 !
- Sẽ là không đủ nếu chỉ thay đổi một toán tử trong một biểu thức lớn
  - Phải lưu ý đến tất cả các phép toán riêng rẽ sẽ được thực hiện

Ép kiểu

- Ép kiểu các biến
  - Có thể thêm ".0" vào các literal để ép phép toán độ chính xác, nhưng với các biến thì sao ?
- static_cast<double> intVar: ép kiểu tường minh hoặc chuyển intVar sang kiểu double
  - VD: doubleVar=static_cast<double>intVar1/intVar2;
  - Ép thực hiện phép chia số thực giữa hai biến nguyên
- Hai cách ép kiểu
  - Ép kiểu ngầm hoặc tự động: ví dụ biểu thức *17/ 5.5* sẽ tự động chuyển 17 thành 17.0
  - Ép kiểu tường minh:
    (double)17/ 5.5 hoặc (double)myInt/ myDouble

Các toán tử viết tắt

- Các toán tử tăng giảm
  - Toán tử tăng, ++

    intVar++; tương đương với
    intVar = intVar + 1;
  - Toán tử giảm, --

    intVar--; tương đương với
    intVar = intVar – 1;
  - Tăng hậu tố: intVar++
    - Sử dụng giá trị hiện tại của biến, sau đó tăng biến
  - Tăng tiền tố: ++intVar
    - Trước hết tăng biến, sau đó sử dụng giá trị mới

Tăng hậu tố vs Tăng tiền tố

- VD 1: Giá trị của *valueProduced* và *n* ?

```
int  n = 2, valueProduced;
valueProduced = 2 * (n ++);
cout << valueProduced << endl;
cout << n << endl;
```

- VD 2: Giá trị của *valueProduced* và *n* ?

```
int  n = 2, valueProduced;
valueProduced = 2 * (++ n);
cout << valueProduced << endl;
cout << n << endl;
```

Vào/ra dữ liệu

- Các đối tượng I/O: *cin, cout, cerr*
- Được định nghĩa trong thư viện <iostream>
- Phải có các dòng này ở đầu chương trình

  #include <iostream>

  using namespace std;
- Tất cả các kiểu dữ liệu đều có thể hiển thị trên màn hình:
  - Biến, hằng, literal, biểu thức
  - cout << numberOfGames << " games played."; sẽ xuất ra giá trị của biến numberOfGames và chuỗi ký tự " games played."

Xuống dòng khi xuất dữ liệu

- Tạo dòng mới:
  - Sử dụng chuỗi escape "\n"
    - VD: *cout << "Hello World\n";*
  - Sử dụng đối tượng *endl*
    - VD: *cout << "Hello World" << endl;*
  - Hai ví dụ cho kết quả giống nhau

Định dạng giá trị in ra

- Giá trị in ra có thể không như mong muốn. VD:
  cout << "The price is $" << price << endl;
  - Nếu biến price có giá trị 78.5, màn hình sẽ hiển thị:
    - The price is $78.500000 hoặc
    - The price is $78.5
- Quy định kích thước phần thập phân:
  - cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
  - Lệnh *cout* ở VD trên cho kết quả là:
    - The price is $78.50

Xuất ra lỗi

- Sử dụng đối tượng *cerr*
  - Làm việc tương tự như cout
  - Cung cấp cơ chế để phân biệt giữa xuất dữ liệu thông thường và xuất ra lỗi
- Chuyển hướng luồng xuất ra
  - Hầu hết các hệ thống cho phép cout và cerr được chuyển hướng sang các thiết bị khác. VD: máy in, file, …

Sử dụng đầu vào cin

- cin dùng để nhập dữ liệu, cout dùng để xuất dữ liệu
- Khác nhau:
  - Toán tử ">>" hướng theo chiều ngược lại
  - Tên là *cin* thay cho *cout*
  - Chỉ được nhập dữ liệu cho biến
- cin >> num;
  - Dừng màn hình đợi nhập dữ liệu vào
  - Giá trị nhập vào được gán cho biến *num*

Yêu cầu nhập giá trị

- Luôn luôn yêu cầu người dùng nhập giá trị

```
cout << "Enter number of dragons: ";
cin >> numOfDragons;
```

- Nếu không có "\n" trong *cout* con chạy đợi nhập dữ liệu ở trên cùng với lời nhắc

```
Enter number of dragons: _____
```

- Tất cả *cin* nên có lời nhắc *cout*

Phong cách lập trình

- Mục tiêu: làm cho chương trình dễ đọc và thay đổi
- Chú thích trong C++, có 2 cách:
  1. *// cau chu thich cho mot dong*
  2. */* doan chu thich */*
- Quy ước đặt tên trong C++
  - Tên phải có ý nghĩa
  - Tên hằng được viết toàn bộ
    VD: NUMBER_OF_STUDENTS
  - Tên biến được viết theo cách lowerToUpper
    VD: numberStudent

Thư viện

- Các thư viện chuẩn của C++
- # include <tên_thư_viện>
  - Khai báo này thêm nội dung của file thư viện vào chương trình của bạn
- C++ cung cấp sẵn rất nhiều thư viện: xử lý vào/ra, toán học, chuỗi ký tự, …

Không gian tên

- Không gian tên xác định một tập các tên được định nghĩa
- Hiện tại sử dụng không gian tên *std*
  - Nó có tất cả các định nghĩa thư viện chuẩn chúng ta cần
- Ví dụ:

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

- Thay vì phải viết std::cin, chúng ta chỉ cần viết cin

Tóm tắt

- C++ là phân biệt chữ hoa, chữ thường
- Nên đặt các tên (biến và hằng số) có ý nghĩa
- Các biến phải được khai báo trước khi sử dụng, và nên được khởi tạo
- Độ chính xác phép toán phụ thuộc toán hạng bậc cao nhất
- *#include* các thư viện chuẩn của C++ khi cần thiết
- Các đối tượng cin, cout, cerr
- Sử dụng các chú thích khi lập trình giúp chương trình dễ hiểu