Ngôn ngữ lập trình

Bài 1:
Giới thiệu về C++

Giảng viên: Lê Nguyễn Tuấn Thành
Email: thanhlnt@tlu.edu.vn

Bộ Môn Công Nghệ Phần Mềm – Khoa CNTT
Trường Đại Học Thủy Lợi

Nội dung

1. Giới thiệu C++
   - Nguồn gốc, Thuật ngữ
2. Biến (variable), Literal, Hằng số (constant)
3. Biểu thức logic
4. Input/Output
5. Phong cách lập trình
6. Thư viện và Namespace

Bài giảng có sử dụng hình vẽ trong cuốn sách "*Absolute C++*. W. Savitch, Addison Wesley, 2002"

1. Giới thiệu C++

- Thế nào là Ngôn ngữ, Ngôn ngữ lập trình?
- Nguồn gốc (Sự tiến hóa của ngôn ngữ lập trình)
  - Ngôn ngữ bậc thấp: *Assembly*
  - Ngôn ngữ bậc cao: *C, C++, FORTRAN, COBOL, …*
  - Ngôn ngữ hướng đối tượng (Object-Oriented Programming): *C++, Java, …*
- Một số thuật ngữ trong C++
  - Chương trình (*Program*),
  - Hàm (*Function*),
  - Thư viện (*Library*)
  - Input/Output (*IO*)

Ví dụ chương trình C++ (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  }
```

Ví dụ chương trình C++ (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.

2. Biến (variable) trong C++

- Biến (variable) trong C++
  - Một vùng bộ nhớ để lưu trữ dữ liệu cho một chương trình
  - PHẢI KHAI BÁO tất cả dữ liệu trước khi sử dụng trong chương trình
- Cách đặt tên biến trong C++
  - Từ khóa (keyword) hoặc từ dành riêng <> Tên biến
  - Tên biến phân biệt chữ hoa chữ thường
  - Tên biến nên là những tên có nghĩa (theo chuẩn)
- Toán tử số học: +, -, *, /, %, ++, --

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

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 bằng một câu khai báo
  - Nếu không khai báo, một biến sẽ có giá trị "*undefined*"!
    - *int myVar = 0;*
- Gán dữ liệu trong lúc chạy
  - ***Lvalue*** (phía bên trái) & ***Rvalue*** (phía bên phải)
    - **Lvalue** phải là các biến
    - **Rvalue** có thể là bất kỳ biểu thức nào
    - Ví dụ: *distance = rate * time;*
      - Lvalue là *distance*
      - Rvalue 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);`

- Post-Increment: *count* ++
  - Dùng giá trị hiện tại của biến trước, sau đó mới tăng giá trị thêm 1
- Pre-Increment: ++*count*
  - Tăng giá trị hiện tại của biến thêm 1 trước, sau đó mới dùng giá trị mới này

Gán dữ liệu: ký hiệu viết tắt (2/2)

- Câu hỏi 1: giá trị của *valueProduced* và *n* ?

```
1. int  n = 2, valueProduced;
2. valueProduced = 2 * (n ++);
3. cout << valueProduced << endl;
4. cout << n << endl;
```

- Câu hỏi 2: giá trị của *valueProduced* và *n* ?

```
1. int  n = 2, valueProduced;
2. valueProduced = 2 * (++ n);
3. cout << valueProduced << endl;
4. cout << n << endl;
```

Những quy tắc khi gán dữ liệu

- Tính tương thích của dữ liệu
  - Quy tắc chung là không gán một kiểu dữ liệu cho một biến thuộc kiểu dữ liệu khác
  - ***intVar = 2.85***; (intVar là một biến kiểu *int*) => giá trị 2 được gán cho biến *intVar*
    - Chỉ phần integer là phù hợp, vì thế nó được gán cho biến intVar
    - Được gọi là "*gán dữ liệu ngầm (implicit)*" hoặc "*tự động chuyển đổi dữ liệu (automatic type conversion)*"

Chuỗi escape

- Cấu trúc: **\<ký_tự>**
- Thông báo với trình biên dịch đó một chuỗi ký tự đặc biệt

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.)

\'	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

Literal & HẰNG SỐ (CONSTANT)

- Literal
  - 2, 5.85, "A", "Hello World"
  - Không thay đổi trong chương trình
- Hằng số kiểu literal cung cấp ít ý nghĩa. Ví dụ: số 24 không diễn đạt được thông tin gì
- Hằng số được đặt tên (*Named Constant* hoặc *Declared Constant*) cung cấp ý nghĩa muốn diễn đạt. Ví dụ:
  - *constant int NUMBER_OF_STUDENTS = 24;*

Named constant

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 trong phép tính (1/2)

- RẤT QUAN TRỌNG. Biểu thức trong C++ có thể được tính toán không như bạn mong đợi
- Toán hạng (operand) có thứ tự cao nhất sẽ quyết định độ chính xác (int, float …) được thực hiện
- Ví dụ:
  - 17 / 5 => 3
    - Cả hai toán hạng đều là số nguyên (integer) => Độ chính xác theo kiểu số nguyên được thực thi !
  - 17.0 / 5 => 3.4
    - Toán hạng có thứ tự cao nhất là kiểu double (17.0) => Độ chính xác theo kiểu double được thực thi !
  - int var1 = 1; int var2 = 2;
    - Câu hỏi: Giá trị của var1/var2 ?

Độ chính xác trong phép tính (2/2)

- Phép tính được thực hiện từng bước từ trái qua phải
  - 1 / 2 / 3.0 / 4 : Thực thi theo 3 bước
    1. 1 / 2 => 0
    2. 0 / 3.0 => 0.0
    3. 0.0 / 4 => 0.0

Độ ưu tiên của toán tử (1/4)

Display 2.3 **Precedence of Operators**

`::`	Scope resolution operator
`.`	Dot operator
`->`	Member selection
`[]`	Array indexing
`( )`	Function call
`++`	Postfix increment operator (placed after the variable)
`--`	Postfix decrement operator (placed after the variable)
`++`	Prefix increment operator (placed before the variable)
`--`	Prefix decrement operator (placed before the variable)
`!`	Not
`-`	Unary minus
`+`	Unary plus
`*`	Dereference
`&`	Address of
`new`	Create (allocate memory)
`delete`	Destroy (deallocate)
`delete[]`	Destroy array (deallocate)
`sizeof`	Size of object
`( )`	Type cast

Highest precedence (done first)

Độ ưu tiên của toán tử (2/4)

*	Multiply
/	Divide
%	Remainder (modulo)
+	Addition
–	Subtraction
<<	Insertion operator (console output)
>>	Extraction operator (console input)

Lower precedence (done later)

Độ ưu tiên của toán tử (3/4)

Display 2.3 Precedence of Operators

All operators in part 2 are of lower precedence than those in part 1.

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal
!=	Not equal
&&	And
||	Or

Độ ưu tiên của toán tử (4/4)

=	Assignment
+=	Add and assign
−=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulo and assign
? :	Conditional operator
throw	Throw an exception
,	Comma operator

Lowest precedence (done last)

Ví dụ độ ưu tiên của toán tử

- Toán tử số học có độ ưu tiên cao hơn toán tử logic
  - x + 1 > 2 || x + 1 < -3 tương đương với (x + 1) > 2 || (x + 1) < -3
- Số nguyên được sử dụng như các giá trị boolean
  - Giá trị khác 0: true
  - Giá trị 0: false

Ép kiểu
(Type casting)

- Ép kiểu cho các biến
  - Có thể thêm ".0" vào các literal để ép buộc thay đổi độ chính xác, nhưng với các biến thì sao ?
- *static_cast<double> intVar*: ép hoặc chuyển đổi *intVar* sang kiểu *double*
  - Đây là các chuyển đổi kiểu giá trị tường minh (explicit)
- Hai cách ép kiểu
  - Ép kiểu ngầm (implicit) 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

3. Biểu thức logic

- Toán tử logic trong C++
  1. Toán tử AND (&&)
  2. Toán tử OR (||)
  3. Toán tử NOT (!): phủ định
- Toán tử quan hệ (so sánh) trong C++

Display 2.1 Comparison Operators

MATH SYMBOL	ENGLISH	C++ NOTATION	C++ SAMPLE	MATH EQUIVALENT
=	Equal to	==	x + 7 == 2*y	x + 7 = 2y
≠	Not equal to	!=	ans != 'n'	*ans* ≠ *'n'*
<	Less than	<	count < m + 3	*count* < m + 3
≤	Less than or equal to	<=	time <= limit	*time* ≤ *limit*
>	Greater than	>	time > limit	*time* > *limit*
≥	Greater than or equal to	>=	age >= 21	*age* ≥ 21

Giá trị của biểu thức logic

- Kiểu dữ liệu bool. Hai giá trị: *true*, *false* là các hằng số được định nghĩa trước

Display 2.2 **Truth Tables**

AND

Exp_1	*Exp_2*	*Exp_1 && Exp_2*
true	true	true
true	false	false
false	true	false
false	false	false

OR

Exp_1	*Exp_2*	*Exp_1 || Exp_2*
true	true	true
true	false	true
false	true	true
false	false	false

NOT

Exp	*!(Exp)*
true	false
false	true

4. Input/Output

- Đối tượng I/O: *cin, cout, cerr*
- Thư viện <*iostream*>

  #include <iostream>

  using namespace std;
- *cout << numberOfGames << " games played.";* sẽ in ra màn hình giá trị của biến *numberOfGames* và chuỗi ký tự " *games played.*“
- In dòng mới:
  - Sử dụng chuỗi “\n” (newline): *cout << "Hello World\n";*
  - Sử dụng đối tượng *endl*: *cout << "Hello World" << endl;*

Định dạng giá trị in ra

- *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
- Chỉ định kích thước phần thập phân: *cout.precision(2);* => The price is $78.50

Error output

- Sử dụng đối tượng *cerr*
  - Tương tự như *cout*
  - Cung cấp cơ chế để phân biệt màn hình bình thường và màn hình lỗi
- Chuyển hướng luồng in ra
  - Hầu hết các hệ thống cho phép *cout* và *cerr* được chuyển hướng sang các thiết bị khác (máy in, file, …)

Input sử dụng cin

- *cin* dùng để nhập dữ liệu cho các biến <> *cout* dùng để in dữ liệu ra
- *cin >> var*
  - Dấu nhắc trên màn hình đợi nhập dữ liệu vào
  - Giá trị nhập vào được gán cho biến *var*

5. Phong cách lập trình

- Mục tiêu: tạo chương trình dễ đọc và thay đổi
- Chú thích (comment) trong C++, có 2 cách:
  1. *// câu chú thích*
  2. */* đoạn chú thích */*
- Một vài quy ước đặt tên trong C++
  - Quy tắc: tên phải có ý nghĩa
  - Sử dụng CHỮ_HOA cho các hằng số (ví dụ: NUMBER_OF_STUDENTS)
  - Đặt tên biến theo định dạng **lowerToUpper** (ví dụ: numberStudent), có thể thêm kiểu dữ liệu vào đầu tên biến (ví dụ: *iNumberStudent, fCount*, …)

6. Thư viện

- *# include <tên_thư_viện>*
- C++ cung cấp sẵn rất nhiều thư viện: xử lý vào/ra (Input/Output), tính toán (*math*), chuỗi ký tự (*string*) …

Namespace

- Namespace xác định một tập các tên được định nghĩa
- Ví dụ:

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

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

Tóm tắt

- C++ là ngôn ngữ lập trình 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 tính toán phụ thuộc toán hạng có thứ tự cao nhất
- #include các thư viện khi cần thiết
- Đối tượng *cin, cout, cerr*
- Sử dụng các chú thích khi lập trình giúp chương trình dễ hiểu

Giáo trình Tham khảo

- **Giáo trình chính: W. Savitch, *Absolute C++*, Addison Wesley, 2002**
- Tham khảo:
  - A. Ford and T. Teorey, *Practical Debugging in C++*, Prentice Hall, 2002
  - Nguyễn Thanh Thủy, *Kĩ thuật lập trình C++*, NXB Khoa học và Kĩ Thuật, 2006