Ngôn ngữ lập trình C

Bài 4. Biểu thức, nhập và xuất theo định dạng

Soạn bởi: TS. Nguyễn Bá Ngọc

2021

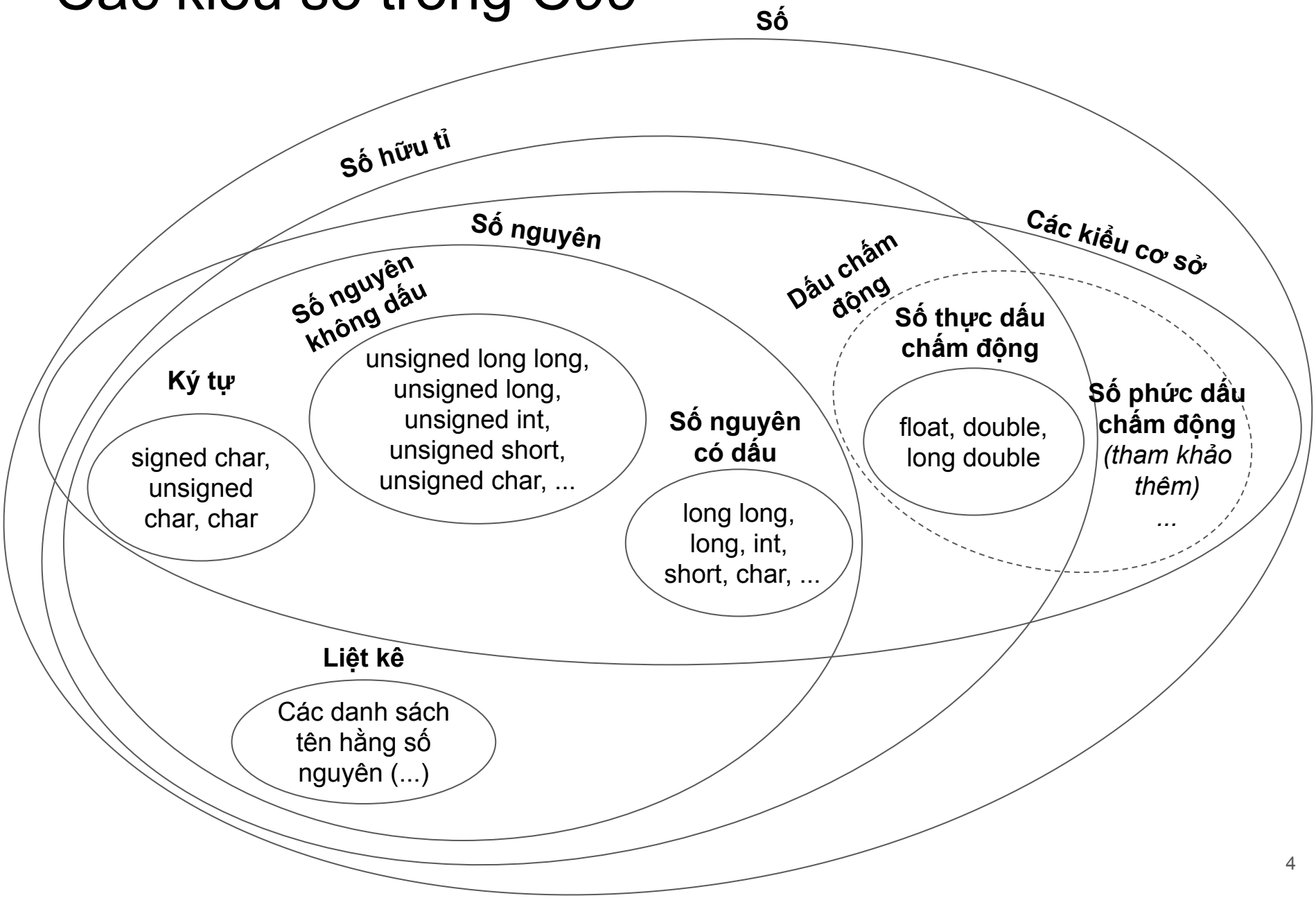
Nội dung

- Biểu thức
- Nhập và xuất theo định dạng

Nội dung

- Biểu thức
- Nhập và xuất theo định dạng

Các kiểu số trong C99



Tổng quan về biểu thức trong C

- Trong C định danh, hằng giá trị, và biểu thức thu được bằng cách đặt một biểu thức trong cặp dấu ngoặc đơn () là những biểu thức cơ bản.
  - Biểu thức dạng (E) có kiểu và giá trị giống như biểu thức E.
- Các biểu thức có thể được kết hợp bằng các toán tử theo quy tắc để tạo thành các biểu thức lớn hơn.
  - Trong chương này chúng ta sẽ học một số toán tử thông dụng cho kiểu số. Các toán tử khác sẽ được khám phá dần theo tiến trình học.
- Khác với khái niệm toán tử trong toán học, toán tử trong C có thể có hiệu ứng phụ (ví dụ làm thay đổi giá trị của biến).
  - Chúng ta chỉ nên thay đổi giá trị của 1 biến tối đa 1 lần trong 1 biểu thức.
  - (*Tham khảo thêm về đơn vị tuần tự/sequence point*).

Biểu thức và câu lệnh

- Biểu thức có thể là thành phần của câu lệnh, trong trường hợp đó biểu thức được tính khi câu lệnh được thực hiện.
- Câu lệnh có thể chỉ bao gồm biểu thức và dấu ; - chúng ta gọi các câu lệnh như vậy là câu lệnh tính biểu thức.
  - Câu lệnh cũng có thể chỉ bao gồm dấu ; (được gọi là câu lệnh null - không làm gì cả, nhưng cần thiết để đáp ứng yêu cầu cú pháp trong một số trường hợp).
  - Trong chương này chúng ta chủ yếu làm việc với câu lệnh tính biểu thức.
  - *(Các câu lệnh khác cũng sẽ được khám phá sau theo tiến trình học phần).*

Các toán tử cộng, trừ

Quy tắc ép kiểu phổ thông được áp dụng với các toán hạng.
Khi sử dụng với các toán hạng kiểu số chúng ta có:

- Toán tử cộng (+)
  - Cho kết quả là tổng của 2 toán hạng.
  - Ví dụ: 1 + 2.0 => 3.0 kiểu double
- Toán tử trừ (-)
  - Cho kết quả là hiệu của 2 toán hạng.
  - Ví dụ: 3 - 1 => 2 kiểu int

Biểu thức cộng và biểu thức trừ có toán hạng kiểu con trỏ có ý nghĩa khác và sẽ được học sau.

Các toán tử dấu

Quy tắc nâng kiểu số nguyên được áp dụng. Toán hạng phải có kiểu số:

- Toán tử dấu +:
  - Cho kết quả là giá trị của toán hạng (sau khi chuyển kiểu)
  - Ví dụ:
    signed char ch = 3;
    +ch => 3 kiểu int
- Toán tử dấu - :
  - Cho kết quả là giá trị của toán hạng (đã chuyển kiểu) sau khi đảo dấu
  - Ví dụ:
    signed char ch = -5;
    -ch => 5 kiểu int.

Các toán tử nhân, chia, và phần dư

Các toán hạng phải có kiểu số. Với toán tử % các toán hạng phải có kiểu số nguyên. Quy tắc ép kiểu phổ thông được áp dụng cho các toán hạng:

- Toán tử nhân (*): Kết quả là tích của 2 toán hạng.
  - Ví dụ: 2 * 1.5 = > 3.0 kiểu double
- Toán tử chia (/):
  - Nếu có toán hạng kiểu số thực dấu chấm động thì kết quả là thương của 2 toán hạng, nếu ngược lại (khi chia 2 số nguyên) thì kết quả là phần nguyên của phép chia.
  - Chia cho 0 là hành vi không xác định.
  - Ví dụ: 3/2 => 1 kiểu int; 3.0/2 => 1.5 kiểu double
- Toán tử phần dư (%):
  - Chỉ áp dụng cho các toán hạng có kiểu số nguyên, kết quả là phần dư của phép chia. Ví dụ, 15 % 6 => 3 kiểu int.

Phần nguyên và phần dư trong phép chia

Với a, b là các số nguyên trong đó b != 0 chúng ta có:

a = (a / b) * b + a % b, bên cạnh đó:

- Trong C89 kết quả phép chia với 2 toán hạng là các số nguyên trong đó có toán hạng là số âm phụ thuộc vào triển khai (có thể được làm tròn lên hoặc làm tròn xuống).
  - 7 / (-3) có thể cho kết quả -2 hoặc -3 tùy theo triển khai.
- Trong C99 kết quả phép chia trong tình huống tương tự được quy ước làm tròn về phía giá trị 0 (bằng kết quả chia thông thường, và là hành vi xác định), ví dụ:
  - 7/(-3) => -2 kiểu int, và 7 % (-3) => 1 kiểu int
  - (-7)/3 => -2 kiểu int, và (-7) % 3 => -1 kiểu int.

Lưu ý chung đối với các toán tử trên dãy bits

Các toán tử trên dãy bits bao gồm: >>, <<, &, |, ^, ~. Các chi tiết sẽ được cung cấp trong các trang tiếp theo.

Các toán hạng của các toán tử trên dãy bits phải có kiểu số nguyên: Có thể là số nguyên không dấu hoặc số nguyên có dấu. Tuy nhiên *chỉ nên áp dụng toán tử trên dãy bits với toán hạng có kiểu không dấu. Không nên áp dụng toán tử trên dãy bits với toán hạng thuộc kiểu có dấu, đặc biệt là các toán hạng có giá trị là các số âm do nhiều hành vi bất định và hành vi phụ thuộc triển khai có thể phát sinh khi thay đổi giá trị của các toán hạng đó.*

Các toán tử dịch chuyển dãy bits

Các toán hạng phải có kiểu số nguyên. Quy tắc nâng kiểu số nguyên được áp dụng cho các toán hạng.

- Toán tử dịch chuyển dãy bits sang trái: <<
- Toán tử dịch chuyển dãy bits sang phải: >>
- Giả sử sau khi nâng kiểu thì vế trái có kiểu T được biểu diễn bằng n bits:
  - Nếu toán hạng ở vế phải có giá trị >= n thì hành vi là không xác định.
  - (*Biểu thức được thực hiện tương tự như đã mô tả trong chương 2 với dãy n bits, kết quả có kiểu T*).

Các toán tử AND, OR, và XOR trên dãy bits

Các toán hạng phải có kiểu số nguyên. Quy tắc chuyển kiểu phổ thông được áp dụng.

- Toán tử và/AND trên dãy bits: &
- Toán tử hoặc/OR trên dãy bits: |
- Toán tử hoặc loại trừ/XOR trên dãy bits: ^
- Giả sử sau khi chuyển kiểu các toán hạng có kiểu T được biểu diễn bằng n bits:
  - (*Biểu thức được thực hiện tương tự như đã mô tả trong chương 2 với dãy n bits, kết quả có kiểu T)*

Toán tử đảo dãy bits (~)

Toán hạng phải có kiểu số nguyên. Quy tắc nâng kiểu số nguyên được áp dụng.

- Giả sử sau khi áp dụng quy tắc nâng kiểu, toán hạng có kiểu T được biểu diễn bằng n bits:
  - *(Biểu thức được thực hiện tương tự như đã mô tả trong chương 2 với dãy n bits, kết quả có kiểu T).*

Các toán tử gán và lvalue

Toán hạng ở bên trái toán tử gán phải là *lvalue có thể thay đổi*. *(Động cơ: Cần thay đổi giá trị một vùng nhớ => cần biết địa chỉ vùng nhớ, có thể ghi dữ liệu, tương thích kiểu, v.v.).*

- Toán tử gán lưu giá trị vào đối tượng được xác định bởi toán hạng ở vế trái.
- Kết quả của phép gán là giá trị của toán hạng ở vế trái sau khi được gán giá trị, nhưng không phải lvalue.
- Kiểu của biểu thức gắn là kiểu gốc (unqualified type) của toán hạng ở vế trái.
- Tương tự như các toán tử khác, toán tử gán cũng có thể được sử dụng theo chuỗi:
  - Ví dụ: x = y = (z + 100);
  - *!An toàn với điều kiện x, y, z là các biến không chồng lấn.*
- Không có quy ước thứ tự xử lý các toán hạng.

Các toán tử gán và lvalue(2)

- *lvalue có thể thay đổi* thường dùng nhất chính là biến kiểu số như chúng ta đã gặp cho tới thời điểm này. Các lvalue khác sẽ được khám phá dần theo tiến trình học.
- Ví dụ lvalue:
  - int x, y; // x, y, z đều là các lvalue có thể thay đổi
  - float z; //
- Các toán tử gán được chia thành 2 nhóm:
  - Đơn giản: =, và
  - Kết hợp: *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |=

Các biểu thức gán

- Biểu thức gán đơn giản (với toán tử =) thường được sử dụng để lưu giá trị của một biểu thức vào 1 biến.
  - Giá trị toán hạng ở vế phải được chuyển kiểu thành kiểu của biểu thức gán. (*Tham khảo thêm các ràng buộc về kiểu đối với các toán hạng*).
- Các phép gán đơn giản có dạng e1 = e1 *op* e2, có thể được viết lại bằng các toán tử gán kết hợp theo định dạng e1 *op*= e2, trong đó *op* ∈ {*, /, %, +, -, <<, >>, &, ^, |}
  - Biểu thức gán kết hợp (e1 op= e2) và biểu thức gán đơn giản tương ứng (e1 = e1 op e2) về cơ bản là tương đương.
  - Chỉ có 1 khác biệt nhỏ: Trong biểu thức gán kết hợp thì toán hạng e1 (ở vế trái) chỉ được tính 1 lần.
  - Các điều kiện về kiểu cho e1 op e2 cũng được áp dụng cho biểu thức gán kết hợp e1 op= e2.

Các toán tử tăng và giảm 1

Yêu cầu: Toán hạng là *lvalue có thể thay đổi*.

Có 2 định dạng đối với toán tử tăng 1 (++) và toán tử giảm 1 (--): Hậu tố và tiền tố.

- Định dạng hậu tố:
  - x++: Giá trị của x được tăng lên 1, giá trị biểu thức bằng giá trị ban đầu của x (giống như (tmp = x, x += 1, tmp)).
  - x--: Giá trị của x bị giảm đi 1, giá trị biểu thức bằng giá trị ban đầu của x (giống như (tmp = x, x -= 1, tmp)).
- Các toán tử tăng và giảm 1 dạng tiền tố:
  - ++x: Giá trị của x được tăng lên 1, giá trị biểu thức bằng giá trị của x sau khi tăng (tương đương với (x += 1)).
  - --x: Giá trị của x được giảm đi 1, giá trị biểu thức bằng giá trị của x sau khi giảm (tương đương với (x -= 1)).

Toán tử sizeof

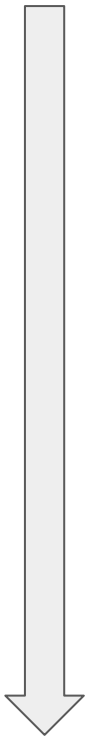
Toán hạng của sizeof có thể là biểu thức hoặc tên kiểu được đặt trong cặp dấu ngoặc đơn.

- sizeof trả về kích thước tính bằng bytes của toán hạng được xác định dựa trên kiểu của toán hạng.
  - *(không áp dụng được sizeof cho kiểu hàm và kiểu chưa hoàn thiện - tham khảo thêm)*.
- Kết quả của toán tử sizeof có kiểu là kiểu số nguyên không dấu được định nghĩa bởi triển khai cụ thể, được đặt tên là size_t và được định nghĩa trong <stddef.h>.
- Một số ví dụ:
  - sizeof(char); // => 1
  - sizeof 10; // Phụ thuộc vào triển khai
  - sizeof 10.2; // => 8
  - sizeof(double); // => 8

Tính biểu thức phức tạp

Các biểu thức con và thứ tự thực hiện các toán tử (khác với thứ tự thực hiện các biểu thức con) trong một biểu thức phức tạp được xác định dựa trên mức ưu tiên và chiều kết hợp của các toán tử.

Ưu tiên cao (thực hiện trước)



Ưu tiên thấp (thực hiện sau)

Tên toán tử	Ký hiệu	Thứ tự
Tăng 1 (hậu tố) Giảm 1 (hậu tố)	++ (x++) -- (x--)	*Trong phạm vi hiện tại hiếm khi cần quan tâm (thường chỉ áp dụng 1 lần cho 1 toán hạng)*
Tăng 1 (tiền tố) Giảm 1 (tiền tố) Toán tử dấu (tiền tố) Toán tử sizeof	++ (++x) -- (--x) +, - (+x, -x) sizeof (sizeof(int))	
Nhân Chia Phần dư	* (x * y) / (x / y) % (x / y)	Trái -> Phải
Cộng Trừ	+ (x + y) - (x - y)	Trái -> Phải
Dịch sang trái Dịch sang phải	<< (x << y) >> (x >> y)	Trái -> Phải
AND theo bit	& (x & y)	Trái -> Phải
XOR theo bit	^ (x ^ y)	Trái -> Phải
OR theo bit	| (x | y)	Trái -> Phải
Gán đơn giản Gán kết hợp	= *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |=	Phải -> Trái

Nội dung

- Biểu thức
- Nhập và xuất theo định dạng

Ví dụ 4-1. Xuất kết quả tính biểu thức

vd4-1.c

```c
#include <stdio.h>

int main() {
  int a = 11, b = 3, c = -5;
  int n, d;
  n = a / b;
  d = a - n * b;

  // %d là đặc tả xuất số nguyên có dấu
  printf("n = %d d = %d (%d)\n",
                n,    d, a % b);
  printf("%d/%d = %d   %d %% %d = %d\n",
            a, c,  a/c,  a,    c, a % c);

  // %u - số nguyên không dấu
  unsigned char ch = 1;
  printf("%u\n", ch << 10);

  // %f - float hoặc double
  double d1 = 11, d2 = 3;
  printf("%f / %f = %f\n", d1, d2, d1 / d2);
  return 0;
}
```

```
bangoc:$gcc -o prog vd4-1.c
bangoc:$./prog
n = 3 d = 2 (2)
11/-5 = -2   11 % -5 = 1
1024
11.000000 / 3.000000 = 3.666667
```

Các biểu thức được xuất tuần tự ở vị trí của các đặc tả xuất.

Bố cục xuất là thông điệp bao gồm các đặc tả xuất sau đó sẽ được áp dụng cho các giá trị của các biểu thức.

Nhập và xuất theo định dạng

Thư viện: <stdio.h>

- Hàm printf: Xuất dữ liệu theo định dạng ra màn hình (stdout)
  - Nguyên mẫu giản lược:

    ```
    int printf (const char *format, ...);
    ```

    format - Tham số thứ nhất là bố cục xuất.
- Hàm scanf: Nhập dữ liệu theo định dạng từ bàn phím (stdin)
  - Nguyên mẫu giản lược:

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

    format - Tham số thứ nhất là bố cục nhập.

Cấu trúc của các đặc tả xuất

- Một số thuật ngữ:
  - Chuỗi bố cục xuất/Template string = Chuỗi định dạng xuất/ Format string
  - Đặc tả xuất/ Output conversion specifications
- Cấu trúc tổng quát của đặc tả xuất

% [stt-tham số $] *các cờ* **độ rộng** [. độ chính xác] định kiểu

hoặc

% [stt-tham số $] *các cờ* **độ rộng** . * [stt-tham số $] định kiểu

Đặc tả xuất bắt đầu bằng dấu % và theo sau là một chuỗi các mô tả. Chỉ có mô tả định kiểu (1 ký tự) là bắt buộc, các mô tả còn lại có thể có hoặc không.