Kỹ Thuật Lập Trình

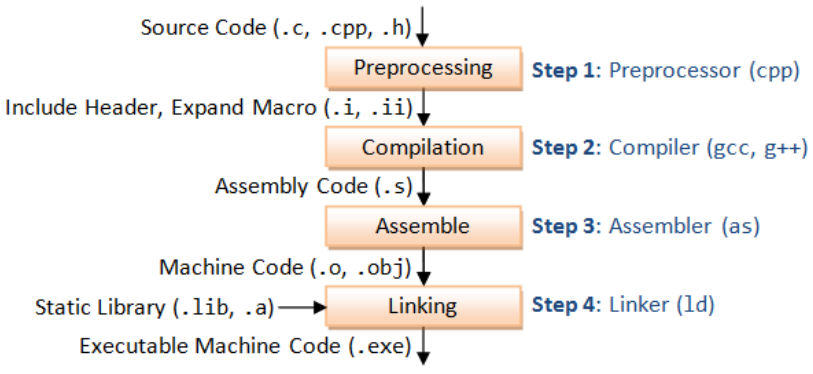
(Ngôn Ngữ Lập Trình C)

Bài giảng số 2

Biến, Giá trị và Kiểu dữ liệu

Nguyễn Hồng Quang

Quy trinh biên dịch (compiler)



Cấu trúc các hệ thống vi xử lý

- Chức năng của *CPU* (central processing unit)
  - Thực thi một câu lệnh tại từng thời điểm
  - Một chuỗi các câu lệnh cấu thành một *chương trình*
- *Bộ nhớ* (memory) lưu trữ thông tin dùng cho CPU
  - Là một chuỗi các *vị trí* được đánh số
  - Mỗi vị trí chứa một đơn vị thông tin
- Thông tin chứa trong CPU và bộ nhớ là một chuỗi các bit: các bit 1 và 0
  - Là dữ liệu dưới dạng nhị phân
  - Tất cả các loại dữ liệu đều có thể biểu diễn dưới dạng nhị phân: các số, thư từ, âm thanh, hình ảnh,…

Ví Dụ

Địa chỉ	Nội dung
0:	01101110
1:	00000000
2:	00000001
3:	10001000
4:	10001001
5:	01110111
6:	00010110

Chương trình (các câu lệnh)

1. *Đặt giá trị 00000001 vào ô nhớ 2*
2. *Đặt giá trị 10001000 vào ô nhớ 3*
3. *Cộng giá trị của ô nhớ 2 với giá trị của ô nhớ 3, kết quả được đặt vào ô nhớ 4*
4. *In giá trị của ô nhớ 2 dưới dạng số nguyên*

Biến

- Nếu lập trình viên phải xử lý thông tin dưới dạng nhị phân hoặc phải nhớ vị trí ô nhớ của dữ liệu... *anh ta có thể bị điên!*
- Ngôn ngữ lập trình cung cấp một phương tiện giúp cho lập trình viên không phải nhớ những chi tiết này
  - *Biến* (variable) là một tên dùng cho một vùng nhớ trong bộ nhớ
  - Biến có *kiểu biến* (type, kiểu dữ liệu); kiểu biến cho phép ta hiểu giá trị của biến theo cách thông thường thay vì hiểu theo giá trị nhị phân
- Câu hỏi: *tại sao các lập trình viên vẫn bị điên đầu*?

Biến Trong C

```
#include <stdio.h>
int main(void)
{

    int  firstOperand;
    int  secondOperand;
    int  thirdOperand;

    firstOperand = 1;
    secondOperand = 2;
    thirdOperand = firstOperand + secondOperand;
    printf("%d", thirdOperand);

    return 0;
}
```

Thứ bạn cần trong mọi chương trình

Khai báo biến (xác định ô nhớ cho các biến)

Các chỉ dẫn cho CPU (các câu lệnh)

Vài Chú Ý

- Mỗi biến trong C được dành riêng một vùng nhớ.
- Tên biến cần được chọn phù hợp nhằm giúp người đọc chương trình hiểu được chức năng của biến trong chương trình.
- Khi tất cả các biến đã có ô nhớ cụ thể, chương trình bắt đầu được thực hiện.
- Tại một thời điểm chỉ có một lệnh được thực hiện, trình tự thực hiện các lệnh phụ thuộc vào vị trí của lệnh trong chương trình.
- Các biến cần được *khởi tạo* trước khi dùng.

Một Ví Dụ Khác

```
#include <stdio.h>
int main(void)
{
  int  rectangleLength;
  int  rectangleWidth;
  int  rectangleArea;

  rectangleLength = 10;
  rectangleWidth  = 3;
  rectangleArea   = rectangleLength * rectangleWidth;
  printf("%d", rectangleArea);

  return 0;
}
```

Tên Biến

- Trong C, *tên* (từ định danh, identifier) cần tuân theo các nguyên tắc sau
  - Dùng kí tự, số và dấu gạch dưới (_)
  - Không bắt đầu với một số
  - Không trùng với *từ dành riêng* (reserved words, từ khóa)
  - Có phân biệt giữa chữ hoa và chữ thường
  - Có thể có độ dài bất kỳ
- *Việc lựa chọn tên rất quan trọng trong việc đọc hiểu chương trình*
  - Thông thường danh từ hoặc chuỗi danh từ mô tả nội dung biến được chọn cho tên biến

Ví Dụ Tên Biến

Đúng	Không đúng	Đúng nhưng...
`rectangleWidth`	`10TimesLength`	`a1`
`rectangle_Width`	`My Variable`	`l`
`rectangle_width`	`int`	`O`

Từ Dành Riêng

- Trong C một số từ nhất định có nghĩa được định sẵn bởi trình biên dịch:
  - **`int`** là một ví dụ
  - Và còn nhiều từ đang chờ bạn
- Những từ đó luôn luôn có nghĩa đặc biệt và không thể được dùng vào các mục đích khác:
  - Không thể dùng làm tên biến
  - Cần được viết chính xác
  - Đôi lúc còn gọi là *từ khóa* (keyword)

Trang 11

Kiểu Dữ Liệu

- Mỗi biến trong C đại diện cho một vùng nhớ trong bộ nhớ.
- Vùng nhớ đó chứa một tập hợp các bit 0 và 1.
- Giá trị thực của tập hợp các bit 0 và 1 trong chương trình C phụ thuộc vào *kiểu dữ liệu* của biến (kiểu biến).
- Ví dụ về 3 kiểu dữ liệu trong C

Số nhị phân	Kiểu biến	Giá trị
01010001	**int**	161
	char	'A'
	double	10.73

Trang 12

Khai Báo Biến

- **`int months;`**
  - Các biến kiểu nguyên đại diện cho các số nguyên
  - 1, 17, -32, 0 Không phải 1.5, 2.0, 'A'
- **`double pi;`**
  - Các biến dấu phẩy động đại diện cho các số thực
  - 3.14, -27.5, 6.02e23, 5.0 Không phải 3
- **`char first_initial, marital_status;`**
  - Các biến kiểu ký tự đại diện cho các ký tự của bàn phím
  - 'a', 'b', 'M', '0' , '9' , '#' , ' ' Không phải "Bill"

Câu Lệnh Gán

- Một *câu lệnh gán* (assignment statement) gán một giá trị cho một biến.
- Phép gán này có thể sử dụng một giá trị cụ thể hay một *biểu thức* (expression).
- CPU sẽ lưu giá trị của biểu thức ở bên trái vào vị trí ô nhớ của biến ở bên phải.

```
int area, length, width;/* khai báo 3 biến */
length= 16;             /* "length có giá trị 16" */
width  = 32;            /* "width có giá trị 32" */
area   = length * width; /* "area có giá trị length*width" */
```

`my_age = my_age+1`

- Đây là một *câu lệnh* (statement), không phải một *phương trình* (equation). Có gì khác biệt?
- Một biến có thể xuất hiện ở cả hai phía (trái và phải) của một câu lệnh.

```
my_age = my_age + 1;
balance = balance + deposit;
```

- Giá trị *cũ* của biến được dùng để tính toán ra giá trị của biểu thức bên phía trái của câu lệnh, trước khi biến thay đổi giá trị.
- *Cách làm này không nên dùng trong toán học.*

Trang 15

Khởi Tạo Biến

- *Khởi tạo biến* là quá trình gán một giá trị nào đó cho biến *lần đầu tiên.*
- Tất cả các quá trình dẫn tới việc thay đổi giá trị của biến đều có thể coi là một cách để khởi tạo biến
  - Câu lệnh gán là một ví dụ
- Quy tắc chung: *biến cần được khởi tạo trước khi sử dụng giá trị của biến đó*
  - Quên khởi tạo biến thường dẫn đến lỗi chương trình
- Biến trong chương trình C *không* tự khởi tạo với một giá trị cụ thể nào.

Trang 16

Khai Báo - Khởi Tạo

```
int main (void)
{
  double income;         /*declaration of income,
                           not an assignment,
                           not an initialization */
  income = 35500.00;     /*assignment to income,
                           initialization of income,
                           not a declaration*/
  printf ("Old income is %f", income);
  income = 39000.00;     /*assignment to income,
                           not a declaration,
                           not an initialization */
  printf ("After raise: %f", income);
}
```


Giải Quyết Vấn Đề / Thiết Kế Chương Trình

- *Xác định* rõ vấn đề.
- *Phân tích* vấn đề.
- Thiết kế *thuật toán* nhằm giải quyết vấn đề.
- *Cài đặt* thuật toán (viết chương trình).

  (nhớ ghi chú thích trong chương trình)
- *Chạy thử* và xác nhận tính đúng đắn của chương trình.

  (chu trình chạy thử - sửa lỗi)
- *Duy trì* và cập nhật chương trình.

Ví Dụ Fahrenheit→Celsius

- Xác định vấn đề
  - Chuyển nhiệt độ từ thang Fahrenheit sang thang Celsius
- Thuật toán (kết quả của quá trình phân tích)
  - Celsius = 5/9 (Fahrenheit - 32)
- Kiểu dữ liệu (kết quả của quá trình phân tích)
  - `double fahrenheit, celsius;`

Ví Dụ F→C (chương trình 1)

```
#include <stdio.h>

int main(void)
{
  double fahrenheit, celsius;

  celsius =  (fahrenheit - 32.0) * 5.0 / 9.0;

  return(0);
}
```

Ví Dụ F→C (chương trình 2)

```
#include <stdio.h>
int main(void)
{
   double fahrenheit, celsius;
   printf("Enter a Fahrenheit temperature: ");
   scanf("%lf", &fahrenheit);
   celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
   printf("That equals %f degrees Celsius.",
     celsius);
   return(0);
}
```

Trang 21

Ví Dụ F→C (chương trình 3)

```
#include <stdio.h>
int main(void)
{
   double fahrenheit, celsius;
   printf("Enter a Fahrenheit temperature: ");
   scanf("%lf", &fahrenheit);
   celsius = fahrenheit - 32.0;
   celsius = celsius * 5.0 / 9.0;
   printf("That equals %f degrees Celsius.",
     celsius);
   return(0);
}
```

Trang 22

Thao Tác Từng Bước

```
celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
```

- Tính giá trị của vế phải
  - Giá trị ban đầu của `fahrenheit` 72.0
  - Trừ đi 32.0 40.0
  - Nhân với 5.0 200.0
  - Chia bởi 9.0 22.2
- Gán giá trị 22.2 cho biến `celsius`
  - Giá trị cũ của `celsius` bị mất

Chú Ý Với Các Ví Dụ

- Bài giảng thường bỏ qua một số chi tiết quan trọng

```
my_age = my_age + 1;
```

- Câu lệnh này *chỉ đúng nếu*
  - `my_age` đã được khai báo trước đó trong chương trình
  - `my_age` có kiểu biến phù hợp (ví dụ kiểu `int`)
  - Xuất hiện ở một vị trí phù hợp
  - Chương trình đã có `int main (void)`
  - …
- Bạn nên sử dụng khả năng suy luận và sáng tạo của chính bạn trong việc chỉ ra những chỗ còn thiếu sót của các ví dụ!

Thuật Ngữ Có Quan Trọng?

- Rất nhiều thuật ngữ được giới thiệu hôm nay
  - *Biến, từ dành riêng, khởi tạo, khai báo, câu lệnh, phép gán,…*
- Bạn có thể viết một chương trình phức tạp mà không cần dùng đến những từ này.
- Nhưng không thể bàn về chương trình của bạn mà không cần đề cập đến chúng.
- Các thuật ngữ này cần được nhớ chính xác và bạn nên có thói quen sử dụng chúng thường xuyên.