# CÁC THÀNH PHẦN TRONG NGÔN NGỮ C

## ĐỊNH DANH VÀ TỪ KHÓA

### Định danh

Định danh là tên được đặt cho các thực thể như biến, hàm, cấu trúc, … Định danh phải là duy nhất. Chúng được tạo ra để đặt một tên duy nhất cho một thực thể để xác định nó trong quá trình thực thi chương trình. Định danh phải khác với từ khóa.

|  |
| --- |
| **VD:** Một số định danh |
| int money;  double accountBalance; |

#### Bảng chữ cái

Mọi ngôn ngữ lập trình đều được xây dựng từ một bộ kí tự nào đó và các quy tắc trên đó để xây dựng các từ, các câu lệnh và cấu trúc chương trình. Ngôn ngữ lập trình C sử dụng bộ ký tự ASCII (American Standard Code for Informations Interchange). Theo chuẩn này, bộ kí tự gồm có 256 kí tự đó là:

* Các chữ cái: A … Z, a .. z
* Các chữ số: 0 1 2 3 4 5 6 7 8 9
* Các dấu phép toán số học: +,-,\*,/,...
* Các ký tự đặc biệt trong lập trình C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Các ký tự đặc biệt trong lập trình C** | | | | |
| , | < | > | . | ~ |
| ( | ) | ; | $ | : |
| % | [ | ] | # | ? |
| ' | & | { | } | " |
| ^ | ! | \* | / | | |

Mỗi kí tự có tương ứng 1 số duy nhất gọi là mã, trong đó có 128 kí tự đầu (có mã từ 0 tới 127) là kí tự cố định và 128 kí tự còn lại (có mã từ 128 tới 255) là các kí tự mở rộng, tức là nó có thể thay đổi tuỳ theo ngôn ngữ mỗi quốc gia sử dụng.

#### Khoảng trắng

Khoảng trắng là thuật ngữ được sử dụng trong C để mô tả khoảng trống, tab, ký tự dòng mới và nhận xét. Khoảng trắng phân tách một phần của một câu lệnh với phần khác và cho phép trình biên dịch xác định vị trí của một phần tử trong một câu lệnh, chẳng hạn như int, kết thúc và phần tử tiếp theo bắt đầu.

|  |
| --- |
| **VD:** Trong câu lệnh sau, có ít nhất một ký tự khoảng trắng giữa int và age để trình biên dịch có thể phân biệt chúng. |
| int age; |
| **VD:** Trong câu lệnh sau, không có ký tự khoảng trắng nào giữa fruit và =, hoặc giữa = và apples, mặc dù bạn có thể thêm một số ký tự nếu muốn tăng khả năng đọc. |
| fruit=apples + oranges; |

#### Quy tắc đặt định danh

Định danh rất quan trọng trong quá trình lập trình, nó không những thể hiện rõ ý nghĩa trong chương trình mà còn dùng để xác định các đối tượng khác nhau trong chương trình.

* Chiều dài tối đa của định danh là 32 ký tự.
* Định danh hợp lệ là một chuỗi ký tự liên tục gồm:
* Ký tự chữ, số và dấu gạch dưới.
* Ký tự đầu phải là chữ hoặc dấu gạch dưới.
* Không được đặt trùng với các từ khóa.
* Định danh nên được đặt tên theo quy tắc Hungarian Notation, CameCase.

**Lưu ý:** Trong ngôn ngữ C có phân biệt chữ hoa, chữ thường.

|  |  |
| --- | --- |
| **VD:** Một số định danh đúng trong ngôn ngữ C. | |
| **Mã chương trình** | **Ý nghĩa** |
| int hocSinh; | Biến hocSinh có kiểu dữ liệu int |
| int Tuoi; | Biến Tuoi có kiểu dữ liệu int |

|  |  |  |
| --- | --- | --- |
| **VD:** Tên biến phân biệt chữ hoa và chữ thường. | | |
| **Mã chương trình** | | **Ý nghĩa** |
| number | Number | Biến number khác biến Number |
| case | Case | case là từ khóa, Case vẫn đúng |

***1. Quy tắc Hungarian Notation***

**VD**: Ta có biến Tuoi có kiểu dữ liệu int, khi sử dụng quy tắc Hungarian Notation bạn thêm chữ i (kí tự đầu của kiểu dữ liệu) vào đầu tên biến Tuoi để trong quá trình lập trình hoặc sau này xem lại, sửa chữa… bạn dễ dàng nhận ra biến Tuoi có kiểu int mà không cần phải di chuyển đến phần khai báo mới biết kiểu của biến này.

|  |
| --- |
| **VD:** Khai báo biến theo quy tắc ***Hungarian Notation*** |
| int ihocSinh;  double dlaiSuat; |

***2. Quy tắc CameCase***

Quy tắc CameCase nghĩa là nếu tên biến có nhiều hơn hai từ thì bắt đầu từ thứ hai trở đi của tên biến sẽ lên hoa.

|  |
| --- |
| **VD:** Khai báo biến theo quy tắc ***CameCase.*** |
| int ihocSinhGioi;  double dlaiSuat; |

### Từ khóa

Từ khóa là các từ dành riêng được sử dụng trong lập trình có ý nghĩa đặc biệt đối với trình biên dịch đã được xác định trước. Từ khóa là một phần của cú pháp và chúng không thể được sử dụng làm hằng số hoặc biến hoặc bất kỳ tên định danh nào khác.

**Lưu ý**: Ngôn ngữ C phân biệt chữ hoa chữ thường, nên tất cả các từ khóa phải được viết bằng chữ thường.

|  |  |
| --- | --- |
| **Mã chương trình** | **Ý nghĩa** |
| int money; | int là một từ khóa cho biết biến money là một biến kiểu int. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Danh sách từ khóa** | | | | | | | |
| auto | continue | double | for | int | signed | struct | void |
| break | do | else | if | long | static | switch | while |
| case | default | enum | goto | register | sizeof | typedef | volatile |
| char | const | extern | float | return | short | union | unsigned |

## COMMENTS TRONG C

Hướng dẫn cách **comment trong C**. Comment trong C được dùng để chú thích nội dung của dòng code viết trong chương trình. Comment trong C sẽ không ảnh hưởng tới kết quả xử lý chương trình, do đó chúng ta có thể tự do viết chúng. Có hai cách **comment trong C** và bạn sẽ học chúng hoàn hảo sau bài viết này.

### Comment trong C là gì

**Comment trong C** hay còn gọi là cách **chú thích trong C** hoặc là **ghi chú trong C** là các dòng code được bỏ qua khi chạy chương trình, nhằm giúp bạn lưu lại các thông tin khi viết chương trình như biến số này dùng làm gì, ai là người tạo và tạo ra lúc nào chẳng hạn. Chỉ cần viết **comment trong C** theo những luật xác định thì dòng comment trong C sẽ không ảnh hưởng tới kết quả chương trình.

Các thông tin cần thiết được lưu giữ bởi **comment trong C** sẽ giúp bạn bảo trì chương trình dễ hơn, cũng như dễ chuyển giao lại dự án cho người khác hoặc là chia sẽ dự án cho nhiều người cùng làm.

### Cách comment trong C

#### Cách comment trên một dòng trong C

Chúng ta sử dụng // để Để comment một dòng trong C. Comment trong C sẽ được tính từ vị trí bắt đầu // cho tới hết dòng.

//dòng comment

Chúng ta có thể bắt đầu comment bằng cách ghi // từ vị trí đầu dòng, hay ở giữa dòng đều được.

|  |
| --- |
| #include <stdio.h>  int main(void){  // người tạo: Xuân Thức  int num;  num = 10;  num = num + 5; // cộng thêm 5 đơn vị  printf("num = %d\n",num);// in ra màn hình  } |

Trong ví dụ trên hai dòng dưới đây chính là comment trong C

|  |
| --- |
| // người tạo: Xuân Thức // cộng thêm 5 đơn vị // in ra màn hình |

#### Cách comment trên nhiều dòng trong C

Để comment trên nhiều dòng trong C, ngoài cách ghi vào đầu từng dòng dấu // như ở cách trên, thì chúng ta có thể dùng một cách đơn giản hơn nữa, đó là viết các dòng chú thích ở giữa cặp dấu /\* và \*/ với cú pháp sau đây:

/\*  
dòng comment 1  
dòng comment 2  
…  
\*/

Chúng ta sử dụng cặp dấu /\* và \*/ để **comment trên nhiều dòng trong C** như ví dụ sau đây:

|  |
| --- |
| #include <stdio.h>  int main(void){  /\*  Ngày viết : 210304  Tác giả : Xuân Thức  \*/   int num = 10;  num = num + 5; /\* cộng thêm 5 đơn vị\*/  printf("num = %d\n",num);// in ra màn hình  } |

Kết quả:

|  |
| --- |
| 15 |

### Sử dụng comment để tạm thời ngừng chạy câu lệnh

Ngoài cách sử dụng **comment trong C** để lưu lại các thông tin cần thiết, bạn cũng có thể dùng comment để ngừng chạy một hoặc nhiều câu lệnh. Cách làm này gọi là **comment out** trong tiếng Anh

Cách sử dụng comment để tạm thời ngừng chạy câu lệnh giống như ví dụ sau đây:

|  |
| --- |
| #include <stdio.h>  int main(void){  int price = 500;  price = price \* 2;  printf("price = %d\n",price); } |

Nếu chạy chương trình trên, chúng ta sẽ có kết quả là:

|  |
| --- |
| price = 1000 |

Nếu chúng ta không muốn chạy dòng lệnh thứ 2 nữa, chúng ta có thể xoá nó. Tuy nhiên nếu chúng ta chỉ tạm thời không muốn chạy dòng lệnh này, hãy comment dòng lệnh thứ hai như sau:

|  |
| --- |
| #include <stdio.h>  int main(void){  int price = 500;  //price = price \* 2;  printf("price = %d\n",price); } |

Khi đó dòng lệnh thứ hai bị ngưng chạy, do đó chỉ có dòng lệnh 1 và 3 được thực thi và kết quả trả về như sau:

|  |  |
| --- | --- |
| price = 500 |  |

Nếu sau đó chúng ta muốn tiếp tục chạy dòng code này, chỉ cần đơn giản bỏ đi dấu // trước nó là xong, mà không cần phải viết lại nó nữa.

### Phím tắt comment trong c

Tùy thuộc vào phần mềm viết code C mà bạn dùng thì chúng ta sẽ có các cách dùng phím tắt khác nhau như dưới đây:

#### Phím tắt comment trong Dev C++

Sau khi chọn các dòng code, chúng ta có thể sử dụng các tổ hợp phím sau để comment trong Dev C++:

* Ctrl +. Chuyển vùng code được chọn thành comment
* Ctrl +, Bỏ commet của vùng code được chọn

#### Phím tắt comment trong Notepad ++

Sau khi chọn các dòng code, chúng ta có thể sử dụng các tổ hợp phím sau để comment trong Notepad ++:

* **Ctrl + K**: Chuyển vùng code được chọn thành comment
* **Ctrl + Q**: Bỏ commet của vùng code được chọn

#### Phím tắt comment trong Visual Studio

Sau khi chọn các dòng code, chúng ta có thể sử dụng các tổ hợp phím sau để comment trong Visual Studio:

* **Ctrl + K + C**: Chuyển vùng code được chọn thành comment
* **Ctrl + K + U**: Bỏ commet của vùng code được chọn

#### Phím tắt comment trong Sublime text 3

Sau khi chọn các dòng code, chúng ta có thể sử dụng các tổ hợp phím sau để comment trong Sublime text 3:

* **Ctrl + /**: Chuyển vùng code được chọn thành comment
* **Ctrl + Shift + /**: Bỏ commet của vùng code được chọn

## BIẾN VÀ HẰNG SỐ

### Công việc của bộ nhớ

Bài viết này có mối liên hệ trực tiếp với bộ nhớ của máy tính.

Con người cũng như máy tính đều cần lưu giữ lại một số cái gì đó, con người chỉ có duy nhất bộ não nhưng trên máy tính thì có nhiều dạng bộ nhớ khác nhau.

|  |
| --- |
| Tại sao máy tính cần nhiều loại bộ nhớ khác nhau? Một bộ nhớ duy nhất không đủ cho máy tính, có phải vậy không? |

Không, thực tế người ta chỉ cần một bộ nhớ có tốc độ lưu nhanh và khả năng chứa lớn (để có thể lưu lại nhiều thứ quan trọng).

Nhưng cho đến thời điểm hiện tại, chúng ta vẫn chưa tạo được những bộ nhớ giống như vậy. Vì các bộ nhớ nhanh thì đắt tiền nên các bộ nhớ được tổ chức thành nhiều cấp, cấp có dung lượng ít thì nhanh nhưng đắt tiền hơn cấp có dung lượng cao hơn. Những bộ nhớ có tốc độ lưu càng nhanh sẽ có dung lượng càng nhỏ.

Vậy máy tính của chúng ta được lắp đặt gồm:

• Những bộ nhớ có tốc độ lưu nhanh nhưng khả năng chứa nhỏ.

• Những bộ nhớ có tốc độ lưu chậm nhưng khả năng chứa lớn hơn rất nhiều.

#### Những khác biệt về bộ nhớ:

Để dễ hiểu, đây là những loại bộ nhớ khác nhau có trong một máy tính được sắp xếp từ nhanh đến chậm:

1. Registers: Bộ nhớ cực nhanh được đặt trực tiếp trong bộ xử lý của máy tính (processor).

2. Memory cache: Làm cầu nối giữa registers và RAM.

3. Main memory (RAM): Là một bộ nhớ mà chúng ta sử dụng thường xuyên nhất.

4. Ổ cứng (Hard Disk Drive): Cái này các bạn biết đến nhiều nhất, người ta thường lưu trữ dữ liệu ở đây.

Những registers chỉ có thể chứa được một vài số, trái ngược hẳn với ổ cứng có thể chứa một số lượng lớn các tập tin.

Từ bây giờ, chúng ta sẽ học về lập trình, và chúng ta thường chỉ làm việc trên RAM nên các bạn cần biết đôi chút về nó. Chúng ta sẽ tìm hiểu cách đọc và lưu các tập tin lên ổ cứng. Còn về Memory cache và registers thì không cần phải chạm đến vì máy tính của bạn sẽ tự làm việc đó.

Cần phải nói thêm một điều quan trọng cuối cùng: chỉ có ổ cứng giữ lại tất cả những gì mà nó chứa. Tất cả các bộ nhớ khác (registers, Memory cache, RAM) đều là những bộ nhớ nhất thời: khi mà bạn tắt máy tính đi thì tất cả dữ liệu trong đó sẽ mất đi.

#### Hình ảnh của RAM

#### Biểu đồ của RAM

|  |  |
| --- | --- |
| **Địa chỉ** | **Giá trị** |
| 1 | 124 |
| 2 | 223 |
| 3 | 113 |
| 4 | 134 |
| …. | … |
| 12355667 | 900.225 |

Như bạn thấy, nó được chia làm 2 cột:

- Một cột địa chỉ (address): địa chỉ là một số cho phép máy tính có thể xác đinh vị trí trong RAM. Nó bắt đầu từ địa chỉ 0 và kết thúc ở địa chỉ 3 448 765 900 126... Giá trị này có thể là rất lớn. Bởi vì nó phụ thuộc vào dung lượng bộ nhớ mà bạn có. Chỉ có thể nói là, bạn có RAM, bạn có thể để vào đó nhiều thứ.

- Và mỗi địa chỉ chứa một giá trị (một số, value): Máy tính của bạn đưa vào RAM những số này để có thể nhớ ngay lập tức. Và người ta chỉ có thể đưa vào một số cho một địa chỉ trong RAM!

Và RAM không thể chứa gì khác ngoài những con số.

|  |
| --- |
| Vậy làm cách nào để chúng ta có thể lưu lại những chữ cái? |

Đó là một câu hỏi thú vị, trên thực tế, đối với máy tính thì những chữ cái cũng là những con số!

Một câu văn chính là một dãy những con số!

Có một bảng viết về sự tương ứng giữ chữ cái và số (bảng mã ASCII), ví dụ số 67 tương ứng với chữ Y

Hãy xem xét một vấn đề đơn giản: nếu máy tính muốn lưu lại giá trị 5), nó sẽ đặt số 5 vào một vị trí nào đó trong bộ nhớ. (Ví dụ tại địa chỉ 3 062 199 902). Sau đó, khi muốn tìm lại giá trị này, máy tính sẽ đến “ô” bộ nhớ n° 3 062 199 902, tại đó nó tìm thấy 5!

Và đó là nguyên tắc hoạt động của bộ nhớ, có thể bạn vẫn còn một chút mập mờ (Đâu là lợi ích của việc đặt một số vào một địa chỉ của bộ nhớ?), bạn sẽ hiểu rõ hơn vấn đề này ở những phần sau.

### Cách khai báo một biến số

Vậy thế nào là một biến số ?

Chỉ đơn giản là một thông tin nhỏ được lưu trữ trong RAM. Chúng ta gọi nó là « biến số » vì nó có thể thay đối trong quá trình thực hiện chương trình.

Các bạn sẽ thấy chương trình của chúng ta sẽ chứa đầy những biến số.

Trên ngôn ngữ C, một biến số có 2 thành phần:

• Một **giá trị**: đó là số mà nó chứa, ví dụ như 5.

• Một **tên gọi**: tên gọi này sẽ giúp ta nhận ra nó.

Trên ngôn ngữ C, chúng ta không cần phải nhớ địa chỉ của biến số, chúng ta chỉ cần chỉ ra tên của biến số. Và bộ dịch (Compiler) sẽ thực hiện việc chuyển đổi giữa chữ và số.

#### Gọi tên một biến số:

Trong ngôn ngữ C, mỗi biến số có một tên gọi, nhưng không phải muốn đặt tên thế nào tùy theo ý thích của bạn cũng được đâu.

Dưới đây là một số nguyên tắc khi đặt tên cho biến số:

• Chúng ta chỉ có thể đặt tên nó bằng những chữ cái viết thường hay viết hoa và những con số (abcABC012…).

• Tên của biến số phải bắt đầu bằng một chữ cái. Chúng ta không được sử dụng khoảng trống « », thay vào đó chúng ta có thể sử dụng kí tự « \_ » (underscore). Đó là kí tự duy nhất không thuộc dạng chữ cái hay số được phép sử dụng.

• Bạn cũng không được phép sử dụng chữ cái mang dấu trọng âm. (ví dụ éèêà…).

Và một điều hết sức quan trọng mà bạn cần phải nắm đó là trong ngôn ngữ C (C++ cũng như thế) có sự khác nhau giữa chữ thường và chữ hoa: chieu\_rong, CHIEU\_RONG và CHieu\_RoNg là tên của 3 biến số khác nhau trong ngôn ngữ C.

Mỗi người có cách thức gọi tên biến số khác nhau.

Có nhiều dạng biến số. Khi bạn yêu cầu nó lưu lại một số, bạn phải nói con số đó thuộc dạng nào. Máy tính của bạn không thể nào có khả năng tự nhận biết chúng, điều đó giúp nó rất nhiều trong việc tự tổ chức, và hạn chế việc sử dụng bộ nhớ một cách vô ích

Khi bạn tạo một biến số, phải ghi nó thuộc **dạng nào**.

Đây là những dạng biến số cơ bản thường dùng trong ngôn ngữ C (sẽ còn một số loại biến số khác trong C++):

|  |  |  |
| --- | --- | --- |
| Type | Dung lượng (octets) | Giá trị chấp nhận |
| Singed char | 1 | **-128** đến **127** |
| int | 2 (on processor 16 bits)  4 (on processor 32 bits) | **-32 768** đến **32 767**  **-2 147 483 648** đến **2 147 483 647** |
| long | 4 | **-2 147 483 648** đến **2 147 483 647** |
| float | 4 | **-3.4\*10-38** đến **3.4\*1038** |
| double | 8 | **-1.7\*10-308** đến **1.7\*10308** |

Từ đó ta có thể khai báo các biến số theo Quy tắc ***Hungarian Notation***

**VD**: Ta có biến Tuoi có kiểu dữ liệu int, khi sử dụng quy tắc Hungarian Notation bạn thêm chữ i (kí tự đầu của kiểu dữ liệu) vào đầu tên biến Tuoi để trong quá trình lập trình hoặc sau này xem lại, sửa chữa… bạn dễ dàng nhận ra biến Tuoi có kiểu int mà không cần phải di chuyển đến phần khai báo mới biết kiểu của biến này.

|  |
| --- |
| **VD:** Khai báo biến theo quy tắc ***Hungarian Notation*** |
| int ihocSinh;  double dlaiSuat; |

### Hằng số

#### **Hằng trong C**

Là một giá trị hằng số không cho phép thay đổi trong quá trình chạy chương trình. Như vậy, chúng ta dùng hằng khi không muốn giá trị bị thay đổi trong suốt thời gian chương trình chạy.

#### **Cách khai báo hằng:**

|  |
| --- |
| **const <kiểu dữ liệu> <tên hằng> = <giá trị>**  // Ví dụ  const int **MAX** = 100;  const float **PI** = 3.14; |

**Lưu ý:**

* Việc khai báo hằng luôn luôn phải đi kèm với khởi tạo giá trị.
* Ngoài việc sử dụng hằng, chúng ta cũng có thể làm điều tương tự với define trong C ( Chúng ta sẽ có 1 bài riêng tìm hiểu về Define).
* Dù sử dụng define hay hằng thì chúng ta vẫn nên viết hoa các chữ cái để đặt tên hằng số.

## KIỂU DỮ LIỆU

### Kiểu dữ liệu

Kiểu dữ liệu của một biến còn thể hiện dung lượng không gian mà nó chiếm trong bộ nhớ và cách diễn giải mẫu bit được lưu trữ.

### Phân loại kiểu dữ liệu

Có 3 loại kiểu dữ liệu khác nhau trong ngôn ngữ C.

|  |  |  |
| --- | --- | --- |
| **STT** | **Loại kiểu dữ liệu** | **Mô tả** |
| 1 | Loại Cơ bản | Số nguyên, số thực, ký tự |
| 2 | Loại bool | Đúng hay sai |
| 3 | Loại void | Giá trị trống |

#### Kiểu dữ liệu cơ bản

Ngôn ngữ C có 4 kiểu dữ liệu cơ bản là: char, interger, float, double

|  |  |
| --- | --- |
| **Kiểu dữ liệu cơ bản** | |
| **Tên** | **Kích thước (Byte)** |
| char | 1 |
| int | 4 |
| float | 4 |
| double | 8 |

Kích thước trong bộ nhớ và miền giá trị của các kiểu dữ liệu còn phụ thuộc vào hệ thống và chương trình dịch tương ứng. Giá trị được đưa ra ở đây là trên hệ thống Windows 64 bit và trình dịch GCC MinGW.

##### Kiểu số nguyên

Trong ngôn ngữ C, có 4 công cụ sửa đổi kiểu dữ liệu. các bổ ngữ này kết hợp với các kiểu dữ liệu chính để phân loại phạm vi sử dụng Tối ưu bộ nhớ lưu trữ.

|  |  |  |
| --- | --- | --- |
| **Các bổ ngữ** | **Mục đích** | **Mô tả** |
| signed | Thay đổi khoảng giá trị của byte. | Có dấu |
| unsigned | Không dấu |
| long | Thay đổi giá trị byte. | 4 byte |
| short | 2 byte |

**Bảng:** Giá trị kiểu dữ liệu số nguyên.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **Mô tả** | **Kích thước** | **Khoảng giá trị** |
| char | Kí tự hoặc số nguyên bé | 1 byte | signed: -  unsigned: 0 |
| short | Số nguyên ngắn | 2 byte | signed: -  unsigned: 0 |
| int | Số nguyên | 4 byte | signed: -  unsigned: 0 |
| long | Số nguyên dài | 4 byte | signed: -  unsigned: 0 |
| long long | Số nguyên cực dài | 8 byte | signed: -  unsigned: 0 |

Ngoài ra, một số kiểu dữ liệu khác như:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **Tên** | **Kích thước** | **Khoảng giá trị** |
| uint8\_t | unsigned char | 1 | 0 |
| uint16\_t | unsigned short  unsigned short int | 2 | 0 |
| uint32\_t | unsigned int  unsigned long | 4 | 0 |

###### 2.1.2 Kiểu số thực

Kiểu số thực được dùng để chứa những số có dấu phẩy động.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **Kích thước** | **Khoảng giá trị** | **Độ chính xác** |
| float | 4 | 1.2E-38 to 3.4E+38 | Độ chính xác 6 con số |
| double | 8 | 2.3E-308 to 1.7E+308 | Độ chính xác 15 con số |
| long double | 16 | 3.4E-4932 to 1.1E+4932 | Độ chính xác 19 con số |

###### 2.1.3 Kiểu ký tự

Kiểu ký tự dùng để chứa những ký tự.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **Mô tả** | **Kích thước** | **Khoảng giá trị** |
| char | Kí tự hoặc số nguyên bé | 1 byte | - hay 0 |
| signedchar | - |
| unsigned char | 0 |

##### 2.3 Kiểu bool

Giá trị 1: là đúng ( true )

Giá trị 0: là sai (false)

##### 2.4 Kiểu void

Kiểu void chỉ định rằng không có giá trị nào. Nó được sử dụng trong ba loại tình huống:

|  |  |
| --- | --- |
| **STT** | **Loại & Mô tả** |
| 1 | ***Hàm trả về dạng void***  Có nhiều hàm khác nhau trong C không trả về bất kỳ giá trị nào hoặc bạn có thể nói rằng chúng trả về giá trị vô hiệu. Một hàm không có giá trị trả về có kiểu trả về là void.  **VD:** void exit (int status); |
| 2 | ***Đối số hàm là void***  Có nhiều hàm khác nhau trong C không chấp nhận bất kỳ tham số nào. Một hàm không có tham số có thể chấp nhận một khoảng trống.  **VD:** int rand (void); |
| 3 | ***Con trỏ void***  Một con trỏ kiểu void \* đại diện cho địa chỉ của một đối tượng, nhưng không phải là kiểu của nó.  **VD:** một hàm cấp phát bộ nhớ void \* malloc (size\_t size); trả về một con trỏ đến void có thể được truyền đến bất kỳ kiểu dữ liệu nào. |

#### 3. Xác định kích thước kiểu dữ liệu

##### 3.1.Toán tử sizeof là gì?

Toán tử sizeof nhận một tham số là bất kỳ kiểu dữ liệu nào và trả về kích thước của kiểu dữ liệu đó.

**Cú pháp:**

|  |
| --- |
| sizeof (Kiểu dữ liệu) |

##### 3.2. Kiểm tra kích thước các kiểu dữ liệu

|  |
| --- |
| **VD:** Xác định kích thước của các kiểu dữ liệu cơ bản trong C. |
| #include<stdio.h>  int main() {      printf("Kich thuoc kieu char: %i\n", sizeof(char));      printf("Kich thuoc kieu int: %i\n", sizeof(int));      printf("Kich thuoc kieu float: %i\n", sizeof(float));      printf("Kich thuoc kieu double: %i\n", sizeof(double));      return 0;  } |

**Kết quả:**

|  |
| --- |
| Kich thuoc kieu char: 1  Kich thuoc kieu int: 4  Kich thuoc kieu float: 4  Kich thuoc kieu double: 8 |

**VD:** Xác định kích thước của các kiểu dữ liệu

|  |
| --- |
| //Xác định kích thước của các kiểu dữ liệu  #include <stdio.h>      int main() {      char c;      short s;      int i;      unsigned int ui;      float f;      double d;      long l;      long long ll;      unsigned long long ull;      // Lấy kích thước từng kiểu dữ liệu//      printf("Size cua kieu char: %i\n",sizeof(c));      printf("Size cua kieu short: %i\n",sizeof(s));      printf("Size cua kieu int: %i\n",sizeof (i));      printf("Size cua kieu unsigned int: %i\n",sizeof (ui));      printf("Size cua kieu float: %i\n",sizeof (f));      printf("Size cua kieu double: %i\n",sizeof (d));      printf("Size cua kieu long: %i\n",sizeof (l));      printf("Size cua kieu long long: %i\n",sizeof (ll));      printf("Size cua kieu unsigned long long: %i\n",sizeof (ull));      return 0;  } |

**Kết quả:**

|  |
| --- |
| Size cua kieu char: 1  Size cua kieu short: 2  Size cua kieu int: 4  Size cua kieu unsigned int: 4  Size cua kieu float: 4  Size cua kieu double: 8  Size cua kieu long: 4  Size cua kieu long long: 8  Size cua kieu unsigned long long: 8 |

**VD:** Xác định kích thước max, min của kiểu dữ liệu số nguyên

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <limits.h>  #include <float.h>  int main() {      printf("\n------------Signed -------------\n");      printf("char      - Bit: %-20lli\n", CHAR\_BIT);      printf("char      - max: %-20lli - min: %-20lli\n",CHAR\_MAX, CHAR\_MIN);      printf("short     - max: %-20lli - min: %-20lli\n",SHRT\_MAX, SHRT\_MIN);      printf("int       - max: %-20lli - min: %-20lli\n",INT\_MAX, INT\_MIN);      printf("long      - max: %-20lli - min: %-20lli\n",LONG\_MAX, LONG\_MIN);      printf("long long - max: %-20lli - min: %-20lli\n",LLONG\_MAX, LLONG\_MIN);      printf("\n------------Unsigned -------------\n");      printf("unsigned char      - max: %-20llu\n",UCHAR\_MAX);      printf("unsigned short     - max: %-20llu\n",USHRT\_MAX);      printf("unsigned int       - max: %-20llu\n",UINT\_MAX);      printf("unsigned long      - max: %-20llu\n",ULONG\_MAX);      printf("unsigned long long - max: %-20llu\n",ULLONG\_MAX);      return 0;  } |

**Kết quả:**

|  |
| --- |
| ------------signed -------------  char - Bit: 8  char - max: 127 - min: 4294967168  short - max: 32767 - min: 4294934528  int - max: 2147483647 - min: 2147483648  long - max: 2147483647 - min: 2147483648  long long - max: 9223372036854775807 - min: -9223372036854775808  ------------unsigned -------------  unsigned char - max: 255  unsigned short - max: 65535  unsigned int - max: 4294967295  unsigned long - max: 4294967295  unsigned long long - max: 18446744073709551615 |

Tệp tiêu đề float.h xác định các macro cho phép bạn sử dụng các giá trị này và các chi tiết khác về biểu diễn nhị phân của số thực trong chương trình của bạn. Ví dụ sau sẽ in không gian lưu trữ được thực hiện bởi một kiểu float và các giá trị phạm vi của nó:

**VD:** Xác định kích thước max, min của kiểu dữ liệu số thực

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <limits.h>  #include <float.h>  int main() {      printf("float       - max: %-20g - min: %-20g\n",FLT\_MAX, FLT\_MIN);      printf("double      - max: %-20g - min: %-20g\n",DBL\_MAX, DBL\_MIN);      printf("long double - max: %-20g - min: %-20g\n",LDBL\_MAX, LDBL\_MIN);      return 0;  } |

**Kết quả:**

|  |
| --- |
| float - max: 3.40282e+038 - min: 1.17549e-038  double - max: 1.79769e+308 - min: 2.22507e-308  long double - max: 3.20528e-317 - min: 3.20528e-317 |

## NHẬP XUẤT DỮ LIỆU

### Xuất dữ liệu trong C

Trong lập trình C, printf () là một trong những chức năng đầu ra chính. Hàm gửi đầu ra được định dạng tới màn hình.

 Ví dụ 1: Đầu ra C

|  |
| --- |
| #include <stdio.h>  int main()  {  // Xuat ten ban ra man hinh  printf("Ngo Xuan Thuc");  return 0;  } |

Đầu ra

|  |
| --- |
| Ngo Xuan Thuc |

Chương trình này hoạt động như thế nào?

* Tất cả các chương trình C hợp lệ phải chứa main () chức năng. Việc thực thi mã bắt đầu từ khi bắt đầu main () chức năng.
* Đây printf () là một chức năng thư viện để gửi đầu ra được định dạng tới màn hình. Hàm in chuỗi bên trong dấu ngoặc kép.
* Để sử dụng printf () trong chương trình của chúng tôi, chúng tôi cần bao gồm stdio.h tệp tiêu đề bằng cách sử dụng câu lệnh #include <stdio.h>.
* Câu return 0; lệnh bên trong hàm main() là "Trạng thái thoát" của chương trình. Nó là tùy chọn.

Ví dụ 2: Đầu ra số nguyên

|  |
| --- |
| #include <stdio.h>  int main()  {  int testInteger = 5;  printf("Number = %d", testInteger);  return 0;  } |

Đầu ra:

|  |
| --- |
| Number = 5 |

Chúng ta sử dụng %d định dạng định dạng để in int các loại. Ở đây, %d bên trong các trích dẫn sẽ được thay thế bằng giá trị của testInterger.

%d sẽ thay thế bởi biến số mà ta đã chỉ ra sau dấu phẩy, trường hợp này là testInterger.

Ví dụ 3: Đầu ra float và double

|  |
| --- |
| #include <stdio.h>  int main()  {  float number1 = 13.5;  double number2 = 12.4;  printf("number1 = %f\n", number1);  printf("number2 = %lf", number2);  return 0;  } |

Đầu ra

|  |
| --- |
| number1 = 13.50000000  number2 = 12.40000000 |

Để in float, chúng tôi sử dụng %ftrình xác định định dạng. Tương tự, chúng tôi sử dụng %lfđể in doublecác giá trị.

Ví dụ 4: Đầu ra Ký tự

|  |
| --- |
| #include <stdio.h>  int main()  {  char chr = 'a';  printf("character = %c", chr);  return 0;  } |

Đầu ra

|  |
| --- |
| character = a |

Còn rất nhiều kí tự đặc biệt khác có thể sử dụng. Nhưng để dễ dàng, lúc này bạn chỉ cần nắm những loại sau:

|  |  |
| --- | --- |
| Format | Type |
| “%d” | int |
| “%ld” | long |
| “%f” | float |
| “%f” | double |

### Nhập dữ liệu trong C

Trong lập trình C, scanf() là một trong những chức năng thường được sử dụng để lấy đầu vào từ người dùng. Chức  năng scanf() đọc đầu vào được định dạng từ đầu vào tiêu chuẩn, chẳng hạn như bàn phím.

Cách sử dụng scanf khá giống với printf. Bạn phải đặt %d hay %lf trong cặp dấu "..." để giải thích với máy tính rằng bạn muốn người dùng đưa vào một số nguyên hay một số thực. Sau đó bạn phải chỉ ra tên của biến số sẽ nhận lấy giá trị đó.

Ví dụ 5: Nhập/xuất số nguyên

|  |
| --- |
| #include <stdio.h>  int main()  {  int testInteger;  printf("Enter an integer: ");  scanf("%d", &testInteger);  printf("Number = %d",testInteger);  return 0;  } |

Đầu ra

|  |
| --- |
| Enter an integer: 4  Number = 4 |

Ở đây, chúng tôi đã sử dụng trình %d xác định định dạng bên trong hàm scanf() để nhận int đầu vào từ người dùng. Khi người dùng nhập một số nguyên, nó sẽ được lưu trữ trong biến testInteger.

Lưu ý rằng chúng tôi đã sử dụng &testInteger bên trong scanf(). Đó là vì &test Integer lấy địa chỉ của testIntegervà giá trị do người dùng nhập được lưu trữ trong địa chỉ đó.

Ví dụ 6: Nhập xuất Float và Double

|  |
| --- |
| #include <stdio.h>  int main()  {  float num1;  double num2;  printf("Enter a number: ");  scanf("%f", &num1);  printf("Enter another number: ");  scanf("%lf", &num2);  printf("num1 = %f\n", num1);  printf("num2 = %lf", num2);  return 0;  } |

Đầu ra

|  |
| --- |
| Enter a number: 12.523  Enter another number: 10.2  num1 = 12.523000  num2 = 10.200000 |

Chúng tôi sử dụng %f và %lf định dạng specifier cho floatvà doubletương ứng.

Ví dụ 7: Nhập xuất kí tự C

|  |
| --- |
| #include <stdio.h>  int main()  {  char chr;  printf("Enter a character: ");  scanf("%c",&chr);  printf("You entered %c.", chr);  return 0;  } |

Đầu ra

|  |
| --- |
| Enter a character: g  You entered g |

Khi một ký tự được nhập bởi người dùng trong chương trình trên, ký tự đó không được lưu trữ. Thay vào đó, một giá trị số nguyên (giá trị ASCII) được lưu trữ.

Và khi chúng tôi hiển thị giá trị đó bằng %cđịnh dạng văn bản, ký tự đã nhập sẽ được hiển thị. Nếu chúng ta sử dụng %dđể hiển thị ký tự, giá trị ASCII của nó được in.

Ví dụ 8: Giá trị ASCII

|  |
| --- |
| #include <stdio.h>  int main()  {  char chr;  printf("Enter a character: ");  scanf("%c", &chr);  // When %c is used, a character is displayed  printf("You entered %c.\n",chr);  // When %d is used, ASCII value is displayed  printf("ASCII value is %d.", chr);  return 0;  } |

Đầu ra

|  |
| --- |
| Enter a character: g  You entered g.  ASCII value is 103. |

### Nhập xuất nhiều giá trị

Chúng ta luôn có thể hiển thị giá trị của nhiều biến số chỉ trong một hàm printf duy nhất. Chỗ này sẽ hiển thị %d và chỗ kia hiển thị %f, tùy theo bạn muốn, sau đó chỉ ra theo thứ tự lần lượt những biến số tương ứng, cách nhau bởi những dấu phẩy.

Đây là cách bạn có thể lấy nhiều đầu vào từ người dùng và hiển thị chúng.

|  |
| --- |
| #include <stdio.h>  int main()  {  int a;  float b;  printf("Enter integer and then a float: ");    // Taking multiple inputs  scanf("%d%f", &a, &b);  printf("You entered %d and %f", a, b);  return 0;  } |

Đầu ra

|  |
| --- |
| Enter integer and then a float: -3  3.4  You entered -3 and 3.400000 |

## DEFINE

### Sự miêu tả

Trong ngôn ngữ lập trình C, chỉ thị #define cho phép định nghĩa các macro trong mã nguồn của bạn. Các định nghĩa macro này cho phép khai báo các giá trị không đổi để sử dụng trong toàn bộ mã của bạn.

Định nghĩa macro không phải là biến và mã chương trình của bạn không thể thay đổi như biến. Bạn thường sử dụng cú pháp này khi tạo các hằng số đại diện cho các số, chuỗi hoặc biểu thức.

### Cú pháp

Cú pháp tạo **Hằng** bằng #define trong ngôn ngữ C là:

|  |
| --- |
| #define CNAME **Giá trị** |
| #define CNAME (**Biểu thức**) |

**CNAME**

Tên của hằng số. Hầu hết các lập trình viên C định nghĩa tên hằng của họ bằng chữ hoa, nhưng đó không phải là yêu cầu của Ngôn ngữ C.

**Giá trị**

Giá trị của hằng số.

**Biểu thức**

Biểu thức có giá trị được gán cho hằng số. Biểu *thức* phải được đặt trong dấu ngoặc đơn nếu nó chứa các toán tử.

**Ghi chú**

* KHÔNG đặt ký tự dấu chấm phẩy ở cuối câu lệnh #define. Đây là một sai lầm phổ biến.

Ví dụ:

Hãy xem cách sử dụng chỉ thị #define với số, chuỗi và biểu thức.

#### Số

Sau đây là một ví dụ về cách bạn sử dụng chỉ thị #define để xác định một hằng số:

|  |
| --- |
| #define AGE 10 |

Trong ví dụ này, hằng số có tên AGE sẽ chứa giá trị là 10

#### Chuỗi

Bạn có thể sử dụng chỉ thị #define để định nghĩa một hằng chuỗi.

Ví dụ:

|  |
| --- |
| #define NAME "TechOnTheNet.com" |

Trong ví dụ này, hằng số được gọi là NAME sẽ chứa giá trị của "TechOnTheNet.com".

Dưới đây là một chương trình C ví dụ nơi chúng tôi xác định hai hằng số này:

|  |
| --- |
| #include <stdio.h>  #define NAME "TechOnTheNet.com"  #define AGE 10  int main()  {  printf("%s is over %d years old.\n", NAME, AGE);  return 0;  } |

Chương trình C này sẽ in như sau:

|  |
| --- |
| TechOnTheNet.com is over 10 years old. |

#### Biểu thức

Bạn có thể sử dụng lệnh #define để xác định một hằng số bằng một biểu thức.

Ví dụ:

|  |
| --- |
| #define AGE (20 / 2) |

Trong ví dụ này, hằng số có tên AGE cũng sẽ chứa giá trị là 10.

Dưới đây là một chương trình C ví dụ trong đó chúng tôi sử dụng một biểu thức để xác định hằng số:

|  |
| --- |
| #include <stdio.h>  #define AGE (20 / 2)  int main()  {  printf("TechOnTheNet.com is over %d years old.\n", AGE);  return 0;  } |

Chương trình C này cũng sẽ in như sau:

|  |
| --- |
| TechOnTheNet.com is over 10 years old. |

## Toán tử

Toán tử là một ký hiệu báo cho trình biên dịch thực hiện các hàm toán học hoặc logic cụ thể. Ngôn ngữ C có nhiều toán tử tích hợp sẵn và cung cấp các loại toán tử sau –

* toán tử số học
* Toán tử quan hệ
* Toán tử logic
* toán tử Bitwise
* Toán tử gán
* Toán tử khác

### Toán tử số học

Toán tử số học thực hiện các phép toán như cộng, trừ, nhân, chia, v.v. trên các giá trị số (hằng và biến).

|  |  |
| --- | --- |
| Toán tử | Ý nghĩa của toán tử |
| + | Phép cộng |
| - | Phép trừ |
| \* | Phép nhân |
| / | Phép chia |
| % | Phép chia lấy dư (phép chia modulo) |

VD1: Toán tử số học

|  |
| --- |
| #include <stdio.h>  int main()  {  int a = 9,b = 4, c;    c = a+b;  printf("a+b = %d \n",c);  c = a-b;  printf("a-b = %d \n",c);  c = a\*b;  printf("a\*b = %d \n",c);  c = a/b;  printf("a/b = %d \n",c);  c = a%b;  printf("Số dư khi a chia cho b = %d \n",c);    return 0;  } |

Kết quả

|  |
| --- |
| a+b = 13  a = 5  a\*b = 36  a/b = 2  Số dư khi a chia cho b = 1 |

Ta nhận thấy ở toán tử +, -, \* kết quả hiển thị giống như chúng ta đều biết. Nhưng ở toán tử / trong tính toán thông thường thì 9/4 = 2.25. Tuy nhiên đầu ra trong chương trình là 2. Đó là bởi vì cả 2 biến a và b đều là các số nguyên nên đầu ra cũng sẽ là số nguyên. Trình biên dịch sẽ bỏ qua các số thập phân và hiển thị 2 thay vì 2.25. Nếu muốn hiển thị 2.25 chúng ta sẽ phải ép kiểu cho kết quả và cái này sẽ nói về sau.

### Toán tử tăng và giảm

Lập trình C có hai toán tử tăng ++và giảm --để thay đổi giá trị của toán hạng (hằng hoặc biến) lên 1 đơn vị.

Số tăng ++ làm tăng giá trị lên 1 trong khi số -- giảm làm giảm giá trị xuống 1. Hai toán tử này là toán tử đơn hạng, nghĩa là chúng chỉ hoạt động trên một toán hạng duy nhất.

Ví dụ 2: Toán tử tăng và giảm

|  |
| --- |
| #include <stdio.h>  int main()  {  int a = 10, b = 100;  float c = 10.5, d = 100.5;  printf("++a = %d \n", ++a);  printf("--b = %d \n", --b);  printf("++c = %f \n", ++c);  printf("--d = %f \n", --d);  return 0;  } |

Kết quả

|  |
| --- |
| ++ a = 11  --b = 99  ++ c = 11,500000  --d = 99,500000 |

Ở đây, các toán tử ++và --được sử dụng làm tiền tố. Hai toán tử này cũng có thể được sử dụng làm hậu tố như a++và a--. Chúng ta sẽ tìm hiểu về chúng sau

### Toán tử gán

Toán tử gán được sử dụng để gán giá trị cho một biến. Toán tử gán phổ biến nhất là =

|  |  |  |
| --- | --- | --- |
| Toán tử | Ví dụ | Được hiểu là |
| = | a = b | a = b |
| += | a += b | a = a + b |
| -= | a -= b | a = a – b |
| \*= | a \*= b | a = a \* b |
| /= | a /= b | a = a / b |
| %= | a %= b | a = a % b |

Ví dụ 3: Toán tử gán

|  |
| --- |
| #include <stdio.h>  int main()  {  int a = 5, c;  c = a; // c is 5  printf("c = %d\n", c);  c += a; // c is 10  printf("c = %d\n", c);  c -= a; // c is 5  printf("c = %d\n", c);  c \*= a; // c is 25  printf("c = %d\n", c);  c /= a; // c is 5  printf("c = %d\n", c);  c %= a; // c = 0  printf("c = %d\n", c);  return 0;  } |

Kết quả

|  |
| --- |
| c = 5  c = 10  c = 5  c = 25  c = 5  c = 0 |

### Toán tử quan hệ

Một toán tử quan hệ kiểm tra mối quan hệ giữa hai toán hạng. Nếu quan hệ là đúng, nó trả về 1; nếu quan hệ là sai, nó trả về giá trị 0.

Các toán tử quan hệ được sử dụng trong [**việc ra quyết định**](https://www.programiz.com/c-programming/c-if-else-statement) và [**các vòng lặp**](https://www.programiz.com/c-programming/c-for-loop) .

|  |  |  |
| --- | --- | --- |
| Toán tử | Ý nghĩa | Ví dụ |
| == | Tương đương với | 5 == 3 (0) |
| > | Lớn hơn | 5 > 3 (1) |
| < | Bé hơn | 5 < 3 (0) |
| != | Khác | 5 != 3 (1) |
| >= | Lớn hơn hoặc bằng | 5 >= 3 (1) |
| <= | Bé hơn hoặc bằng | 5 <= 3 (0) |

Ví dụ 4: Toán tử quan hệ

|  |
| --- |
| #include <stdio.h>  int main()  {  int a = 5, b = 5, c = 10;  printf("%d == %d is %d \n", a, b, a == b);  printf("%d == %d is %d \n", a, c, a == c);  printf("%d > %d is %d \n", a, b, a > b);  printf("%d > %d is %d \n", a, c, a > c);  printf("%d < %d is %d \n", a, b, a < b);  printf("%d < %d is %d \n", a, c, a < c);  printf("%d != %d is %d \n", a, b, a != b);  printf("%d != %d is %d \n", a, c, a != c);  printf("%d >= %d is %d \n", a, b, a >= b);  printf("%d >= %d is %d \n", a, c, a >= c);  printf("%d <= %d is %d \n", a, b, a <= b);  printf("%d <= %d is %d \n", a, c, a <= c);  return 0;  } |

Kết quả

|  |
| --- |
| 5 == 5 is 1  5 == 10 is 0  5 > 5 is 0  5 > 10 is 0  5 < 5 is 0  5 < 10 is 1  5 != 5 is 0  5 != 10 is 1  5 >= 5 is 1  5 >= 10 is 0  5 <= 5 is 1  5 <= 10 is 1 |

### Toán tử logic

Một biểu thức chứa toán tử logic trả về 0 hoặc 1 tùy thuộc vào việc biểu thức cho kết quả đúng hay sai. Toán tử logic thường được sử dụng trong việc ra quyết định [**trong lập trình C.**](https://www.programiz.com/c-programming/c-if-else-statement)

|  |  |  |
| --- | --- | --- |
| Toán tử | Ý nghĩa | Ví dụ |
| && | Logic VÀ. Chỉ đúng nếu tất cả các toán hạng là đúng | Nếu c = 5 và d = 2 thì biểu thức ((c==5) && (d>5)) bằng 0 |
| || | Logic HOẶC. Chỉ đúng nếu một trong hai toán hạng là đúng | Nếu c = 5 và d = 2 thì biểu thức ((c==5) || (d>5)) bằng 1. |
| ! | logic KHÔNG. Chỉ đúng nếu toán hạng là 0 | Nếu c = 5 thì biểu thức !(c==5)bằng 0 |

Ví dụ 5: Toán tử logic

|  |
| --- |
| #include <stdio.h>  int main()  {  int a = 5, b = 5, c = 10, result;  result = (a == b) && (c > b);  printf("(a == b) && (c > b) is %d \n", result);  result = (a == b) && (c < b);  printf("(a == b) && (c < b) is %d \n", result);  result = (a == b) || (c < b);  printf("(a == b) || (c < b) is %d \n", result);  result = (a != b) || (c < b);  printf("(a != b) || (c < b) is %d \n", result);  result = !(a != b);  printf("!(a != b) is %d \n", result);  result = !(a == b);  printf("!(a == b) is %d \n", result);  return 0;  } |

Kết quả:

|  |
| --- |
| (a == b) && (c > b) is 1  (a == b) && (c < b) is 0  (a == b) || (c < b) is 1  (a != b) || (c < b) is 0  !(a != b) is 1  !(a == b) is 0 |

### Toán tử Bitwise ( Ít gặp )

Trong quá trình tính toán, các phép toán như: cộng, trừ, nhân, chia, v.v. được chuyển đổi thành mức bit giúp xử lý nhanh hơn và tiết kiệm điện năng.

Các toán tử bitwise được sử dụng trong lập trình C để thực hiện các hoạt động ở cấp độ bit.

### Toán tử khác

Ngoài các toán tử đã thảo luận ở trên, còn có một số toán tử quan trọng khác bao gồm **sizeof** và **? :** và **,** và **\*** và **&**

|  |  |  |
| --- | --- | --- |
| Toán tử | Ý nghĩa | Ví dụ |
| sizeof() | Trả về kích thước của 1 biến | Sizeof(a) Nếu a là số nguyên sẽ trả về 4 |
| & | Trả về địa chỉ của 1 biến | &a trả về địa chỉ của biến a |
| \* | Con trỏ tới 1 biến | \*a |
| ?: | Biểu thức điểu kiện | Nếu Điều kiện là đúng? thì giá trị X : nếu không thì giá trị Y |
| , | sử dụng để liên kết các biểu thức liên quan với nhau | int a, c = 5, d; |

Ưu tiên toán tử trong C

Thứ tự ưu tiên của toán tử xác định nhóm các thuật ngữ trong một biểu thức và quyết định cách đánh giá một biểu thức. Một số toán tử có quyền ưu tiên cao hơn những toán tử khác; ví dụ, toán tử nhân có độ ưu tiên cao hơn toán tử cộng.

Ví dụ: x = 7 + 3 \* 2; ở đây, x được gán 13, không phải 20 vì toán tử \* có độ ưu tiên cao hơn +, do đó, trước tiên, nó được nhân với 3\*2 rồi cộng thành 7.

Ở đây, các toán tử có mức độ ưu tiên cao nhất sẽ xuất hiện ở đầu bảng, những toán tử có mức độ ưu tiên thấp nhất sẽ xuất hiện ở cuối bảng. Trong một biểu thức, các toán tử có mức độ ưu tiên cao hơn sẽ được đánh giá trước.

Table

Description automatically generated