Chào mừng các bạn trở lại với Series Lập Trình Căn Bản C. Hôm nay chúng ta sẽ học về Các phép toán, toán tử trong ngôn ngữ C.

Trong ngôn ngữ C, chúng ta thường sẽ gặp các loại toán tử sau: Toán tử số học, toán tử quan hệ, toán tử luận lý, toán tử tăng giảm, toán tử gán, toán tử phẩy, toán tử lấy kích thước.

1. Toán tử số học

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Toán tử | Tên | Ví dụ | Kết quả |
| 1 | + | Cộng | 3 + 4 | 7 |
| 2 | - | Trừ | 10.5-3 | 7.5 |
| 3 | \* | Nhân | 5\*5 | 25 |
| 4 | / | Chia | 6/3 | 3 |
| 5 | % | Chia ( lấy dư) | 10%3 | 1 |

Với những bạn chưa được tiếp xúc với bất kì ngôn ngữ lập trình nào, tôi sẽ giải thích về toán tử % ( chia lấy dư).

Khi bạn thực hiện một phép chia, bạn sẽ lấy được 2 thành phần kết quả là thương và số dư. Với phép chia thông thường ( / ) thì kết quả bạn sẽ lấy là Thương ( với biến nguyên), nhưng với phép chia lấy dư ( % ) thì kết quả trả về sẽ là số dư.

**Chú ý:**

1. Phép chia lấy dư (%) yêu cầu 2 số phải là số nguyên.
2. Phép chia thông thường ( / ) bạn có thể lấy kết quả khi sử dụng toàn bộ là kiểu dữ liệu số thực ( float, double,..) hoặc sử dụng ép kiểu ( bạn sẽ được học ở bài sau).

Ví dụ:

|  |
| --- |
| #include <stdio.h>  int main(){  int a = 10, b = 3;  printf("ket qua = %d",a%b); // *ket qua = 1*  return 0;  } |

Khi thực hiện bằng tay thì ta sẽ được 10 / 3 = 3 dư 1, vậy kết quả bài toán sẽ là 1.

Ví dụ tổng thể

|  |
| --- |
| #include <stdio.h>    int main()  {      int a = 5, b = 7;      double c = 4.5, d = 6;        printf("%d + %f = %f \n", a, c, a + c);      printf("%d - %d = %d \n", a, b, a - b);      printf("%d \* %f = %f \n", b, d, b \* d);        /\*Chú ý kết quả các phép chia \*/        printf("%d / %d = %d \n", b, a, b / a);      printf("%f / %d = %f \n", c, a, c / a);      printf("%f / %f = %f \n", c, d, c / d);        printf("%d %% %d = %d \n", b, a, b % a);        return 0;  } |

Các bạn chú ý về phép chia, kết quả sẽ nó sẽ khác nhau, một số bạn muốn biết lí do vì sao thì hãy đọc thêm về bài viết Ép Kiểu Trong Ngôn Ngữ C này.

1. Toán tử Quan hệ ( Toán tử so sánh)

Ngôn ngữ C cung cấp 6 toán tử quan hệ.

Các toán tử quan hệ sẽ có kết quả là 1 ( khi kết quả đúng ) và 0 ( khi kết quả sai).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Toán Tử | Tên | Ví Dụ | Kết quả |
| 1 | == | So sánh bằng | 5 == 5 | 1 |
| 2 | != | So sánh khác | 5 != 5 | 0 |
| 3 | > | So sánh Lớn hơn | 3 > 2 | 1 |
| 4 | < | So sánh Nhỏ hơn | 3 < 2 | 0 |
| 5 | >= | So sánh Lớn hơn hoặc bằng | 7 >= 5 | 1 |
| 6 | <= | So sánh Nhỏ hơn hoặc bằng | 7 <= 7 | 1 |

Chú ý: các toán tử quan hệ thường chỉ dùng cho việc so sánh giá trị, ký tự,… chứ không so sánh xâu ( string). Về cách so sánh xâu thì chúng ta sẽ được học ở những bài sau.

1. Toán tử luận lý ( toán tử logic)

Toán tử logic là những toán tử có kết quả trả về là 1 và 0 ( giống toán tử quan hệ). Nhưng bản thân chúng thường dùng cho việc kết nối các luận lý. Cụ thể như sau

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Toán Tử | Tên | Ví dụ | Kết quả |
| 1 | ! | Phủ định | !(5>4) | 0 |
| 2 | && | Và | 5 > 4 && 5 > 6 | 0 |
| 3 | || | Hoặc | 5 > 4 || 5 > 6 | 1 |

Giải thích các toán tử luận lý:

Phủ định ( ! ): Biến một kết quả từ thành đúng -> sai và ngược lại.

Ta thấy ở ví dụ trên, 5 > 4 thì kết quả sẽ cho ra là 1 ( đúng ), nhưng do việc sử dụng toán tử luận lý nên kết quả trả về là 0 ( sai ).

Và (&& ): Các bạn chú ý ở đây gồm 2 dấu &&, nó sẽ trả về đúng nếu kết quả ở 2 bên đều đúng. Khi một trong 2 bị sai thì kết quả sẽ là sai ( 0 ) .

Hoặc ( || ) : Toán tử này sẽ trả về 1 (đúng ) nếu đúng một trong 2 điều kiện ở 2 bên. Nếu cả 2 kết quả đều sai thì kết quả sẽ trả về 0 ( sai ).

1. Toán tử tăng giảm.

Toán tử tăng ( ++ ) và giảm ( --) thường được thực hiện trên các biến, nó sẽ giúp các biến tăng 1 giá trị hoặc giảm 1 giá trị.

Ví dụ

|  |
| --- |
| #include <stdio.h>  int main(){  int a = 3, b = 2;  a++;  b--;  printf("thay doi lan 1: a = %d, b = %d\n", a, b);  printf("thay doi lan 2: a = %d, b = %d", a++, ++b);  return 0;  } |

Kết quả:

thay doi lan 1: a = 4, b = 1

thay doi lan 2: a = 4, b = 2.

Các bạn chú ý kết quả ở trên, ở lần thay đổi thứ nhất mình đã tăng a lên 1 ( a++) và giảm b xuống 1 ( b--). Khi đó a sẽ tăng lên thành 4 và b sẽ thành 1. Lúc này phép toán sẽ như bình thường.

Ở lần thứ 2, mình đã thực hiện đồng thời 2 việc là thay đổi giá trị và in ra màn hình, lúc này đã có một cái gì đó lạ lùng xảy ra.

Mình đã đặt vị trí của dấu tăng (++) ở những vị trí khác nhau của a và b. Cụ thể: a++ và ++b.

Vì thế kết quả của bài toán cũng đã có sự khác biệt. Giá trị của b khi in ra đã có sự thay đổi trong khi a vẫn giữ nguyên. Vì sao lại vậy?

Vấn đề này được mọi người đưa vào độ ưu tiên của toán tử, phụ thuộc vào tiền tố và hậu tố mà giá trị sẽ thay đổi như thế nào. Mình sẽ giải thích với các bạn như sau:

* Phép toán tăng giảm khi đặt ở sau các biến thì được gọi là Hậu Tố: Lúc này thì kết quả sẽ thực hiện lệnh đang làm trước rồi sau đó mới tăng (giảm) giá trị lên 1.
* Phép toán tăng giảm khi đặt ở trước các biến thì được gọi là Tiền Tố: Lúc này thì kết quả sẽ tăng ( giảm ) giá trị xuống 1 rồi mới thực hiện việc đang làm.

Cụ thể trong bài toán của mình ở lần thay đổi thứ 2 sẽ thực hiện như sau:

Code: printf("thay doi lan 2: a = %d, b = %d", a++, ++b)

* a++: in màn hình giá trị của a trước rồi mới tăng giá trị a lên 1.
* ++b: tăng giá trị của b trước rồi mới in ra giá trị của b.
* Vậy màn hình in ra giá trị sẽ là a = 4, b = 2.

Bạn hãy thử viết thêm một lệnh in ra màn hình lần nữa, kết quả lần tiếp theo sẽ là a = 5 và b = 2. Tại sao vậy?

1. Toán tử gán

Toán tử gán được sử dụng để lưu trữ giá trị cho 1 biến nào đó.

|  |  |  |
| --- | --- | --- |
| Toán tử | Ví dụ | Tương đương với |
| = | x = 5 | Gán 5 cho x |
| += | x += 5 | x = x + 5 |
| -= | x -= 5 | x = x - 5 |
| \*= | x \*= 5 | x = x \* 5 |
| /= | x /= 5 | x = x / 5 |
| %= | x %= 5 | x = x % 5 |

Code:

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

Kết quả:

x = 5 => x = 5  
x += 5 => x = 10  
x -= 5 => x = 5  
x \*= 5 => x = 25  
x /= 5 => x = 5

1. Toán tử phẩy

Nhiều biểu thức có thể được kết nối vào cùng một biểu thức sử dụng toán tử phẩy. Toán tử phẩy yêu cầu 2 toán hạng. Đầu tiên nó ước lượng toán hạng trái sau đó là toán hạng phải, và trả về giá trị của toán hạng phải như là kết quả sau cùng.

#include <stdio.h>

int main()

{

    int m, t;

    m = (t =2, t\*5 + 10);

    printf("t = %d, m = %d\n", t, m);

    return 0;

}

t = 2, m = 20

1. Toán tử lấy kích thước

Ngôn ngữ C cung cấp cho chúng ta toán tử sizeof để tính toán kích thước của các loại dữ liệu. Chúng ta sẽ cần sử dụng chúng khi muốn biết kích thước của kiểu dữ liệu mình đang cần dùng là gì. Cụ thể ví dụ

#include <stdio.h>

int main()

{

    printf("char size = %d byte\n", sizeof(char));

    printf("short size = %d byte\n", sizeof(short));

    printf("int size = %d byte\n", sizeof(int));

    printf("long size = %d byte\n", sizeof(long));

    printf("float size = %d byte\n", sizeof(float));

    printf("double size = %d byte\n", sizeof(double));

    printf("1.55 size = %d byte\n", sizeof(1.55));

    printf("\"Hello\" size = %d byte\n", sizeof("Hello"));

    return 0;

}

Kết quả trả về:

char size = 1 byte  
short size = 2 byte  
int size = 4 byte  
long size = 8 byte  
float size = 4 byte  
double size = 8 byte  
1.55 size = 8 byte  
“Hello” size = 6 byte

1. Độ ưu tiên của các toán tử

Độ ưu tiên các toán tử được thực hiện từ trên xuống dưới theo bảng sau. Trong mỗi hàng lại có độ ưu tiên như ở cột thứ 3.

| **Loại** | **Toán tử** | **Thứ tự ưu tiên** |
| --- | --- | --- |
| Postfix | () [] -> . ++ – – | Trái sang phải |
| Unary | + – ! ~ ++ – – (type)\* & sizeof | Phải sang trái |
| Tính nhân | \* / % | Trái sang phải |
| Tính cộng | + – | Trái sang phải |
| Dịch chuyển | << >> | Trái sang phải |
| Quan hệ | < <= > >= | Trái sang phải |
| Cân bằng | == != | Trái sang phải |
| Phép AND bit | & | Trái sang phải |
| Phép XOR bit | ^ | Trái sang phải |
| Phép OR bit | | | Trái sang phải |
| Phép AND logic | && | Trái sang phải |
| Phép OR logic | || | Trái sang phải |
| Điều kiện | ?: | Phải sang trái |
| Gán | = += -= \*= /= %=>>= <<= &= ^= |= | Phải sang trái |
| Dấu phảy | , | Trái sang phải |

Bài học về toán tử của chúng ta đã kết thúc, các bạn có thể làm một số ví dụ để hiểu rõ hơn về toán tử.

Bài tập: Viết chương trình mô phỏng máy tính casio bỏ túi với các phép toán đơn giản: Cộng, trừ, nhân, chia, chia lấy nguyên.

Tất cả các bài tập sẽ được giải bài viết Giải Bài Tập Series Lập Trình C căn bản.