**Toán Tử - Operator**

**1.Toán Tử Gán**

Toán tử gán (dấu =) được sử dụng trong các bài học trước khi bạn muốn gán giá trị cho biến

Cú pháp :**X = Y**

Ý nghĩa : Gán giá trị của toán hạng Y cho toán hạng X

Ví dụ :

#include <stdio.h>

int main(){

int n = 28;

int m = n;

int p = m;

printf("%d %d %d\n", n, m, p);

return 0;

}

**2. Toán Tử Toán Học**

Toán tử toán học bao gồm các phép toán bạn thường sử dụng : +, -, \*, /, %. Trong lập trình bạn cần chú ý phép nhân sẽ là dấu \*, phép chia là dấu / và phép chia dư là %.

Thứ tự ưu tiên các toán tử này trong biểu thức đó là nhân chia và chia dư trước, cộng trừ sau, cùng mức ưu tiên thì thực hiện từ trái qua phải. Tuy nhiên bạn dùng thêm đóng mở ngoặc vì nó có mức độ ưu tiên cao nhất, sẽ đảm bảo được biểu thức của bạn được tính toán như bạn mong muốn.

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Ý nghĩa** | **Ví dụ** |
| + | Phép cộng | int n = 100 + 200; // n = 300 |
| - | Phép trừ | int n = 200 - 100; // n = 100 |
| \* | Phép nhân | int n = 28 \* 2; // n = 56 |
| / | Phép chia | int n = 50 / 2; // n = 25 |
| % | Phép chia dư | int n = 20 % 3; // n = 2 |

Sau khi học các toán tử toán học này bạn có thể xây dựng được những chương trình tính toán đơn giản rồi.

Ví dụ 1. Nhập chiều dài, rộng và in ra chu vi, diện tích của hình chữ nhật

#include <stdio.h>

int main(){

int dai, rong;

scanf("%d %d", &dai, &rong);

int chuvi = 2 \* (dai + rong);

int dientich = dai \* rong;

printf("%d %d", chuvi, dientich);

return 0;

}

Ví dụ 2 : Nhập bán kính hình tròn, in ra chu vi và diện tích lấy 2 chữ số phần thập phân

#include <stdio.h>

int main(){

double R;

scanf("%lf", &R);

double chuvi = 2 \* 3.14 \* R;

double dientich = 3.14 \* R \* R;

printf("%.2lf %.2lf\n", chuvi, dientich);

return 0;

}

**Chú ý 1** : Chia nguyên và chia lấy phần thập phân

* Nếu bạn lấy 2 số nguyên (int, long long) và chia cho nhau thì phép chia đó là phép chia nguyên, phần thập phân sẽ tự bị loại bỏ
* Nếu bạn chia 2 số thực cho nhau hoặc 1 số nguyên với 1 số thực thì phép chia mới giữ lại phần thập phân.

Ví dụ 3 : Phép chia nguyên

Mặc dù bạn đã dùng số float để lưu thương nhưng kết quả vẫn chỉ là 3.00, vì khi bạn chia a / b thì kết quả đã là 3 rồi.

#include <stdio.h>

int main(){

int a = 10, b = 3;

int thuong1 = a / b;

printf("%d\n", thuong1);

float thuong2 = a / b;

printf("%.2f\n", thuong2);

return 0;

}

Output :

3

3.00

Ví dụ 4 : Phép chia muốn lấy phần thập phân có 2 cách xử lý

* Ép kiểu số chia hoặc bị chia sang số thực
* Nhân số bị chia với 1.0 (mình thích cách này hơn), cách này ngầm biến số bị chia thành 1 số thực trong quá trình tính toán

#include <stdio.h>

int main(){

int a = 10, b = 3;

float thuong1 = (float)a / b;

printf("%.2f\n", thuong1);

float thuong2 = 1.0 \* a / b;

printf("%.2f\n", thuong2);

return 0;

}

Output : 3.33 3.33

**Chú ý 2**: Nhân 2 số nguyên bị tràn dữ liệu

Đầu tiên bạn cần biết khi bạn dùng 2 số int và nhân với nhau thì tích của 2 số này có thể vượt giới hạn của số int, ví dụ a = 106 và b = 106thì tích của 2 số là 1012 vượt giới hạn của số int và sẽ bị tràn. Để xử lý tình huống này bạn có thể lưu 2 số này ở kiểu long long, hoặc ít nhất 1 trong 2 số ở kiểu long long. Cách thứ 2 bạn có thể ép kiểu khi nhân 2 số.

Ví dụ 5 : Nhân 2 số int bị tràn và cách xử lý

#include <stdio.h>

int main(){

int a = 1000000, b = 1000000;

long long tich1 = a \* b;

printf("%lld\n", tich1);

//Ép kiểu

long long tich2 = (long long) a \* b;

printf("%lld\n", tich2);

//Nhân với số 1ll = 1 long long;

long long tich3 = 1ll \* a \* b;

printf("%lld\n", tich3);

return 0;

}

Output :

-727379968

1000000000000

1000000000000

Sau khi học xong toán tử toán học, các bạn có thể sử dụng thêm các toán tử gán kết hợp toán tử toán học :

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Ví dụ** | **Ý nghĩa** |
| += | 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 |

**3.Toán Tử So Sánh**

Các toán tử so sánh thường được sử dụng để kiểm tra điều kiện, các toán tử này sẽ trả về giá trị đúng hoặc sai khi bạn so sánh 2 toán hạng với nhau.

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Ý nghĩa** | **Ví dụ** |
| > | So sánh lớn hơn | a > b |
| >= | Lớn hơn hoặc bằng | a >= b |
| < | So sánh nhỏ hơn | a < b |
| <= | Nhỏ hơn hoặc bằng | a <= b |
| != | So sánh khác | a != b |
| == | So sánh bằng | a == b |

Một vài ví dụ về kết quả của toán tử so sánh :

|  |  |
| --- | --- |
| **Ví dụ** | **Kết quả** |
| 20 > 10 | Đúng |
| 5 <= 15 | Đúng |
| 30 < 50 | Sai |
| 5 <= 5 | Đúng |
| 10 != 10 | Sai |
| 10 == 10 | Đúng |

Ví dụ : Bạn cũng có thể sử dụng code để in ra giá trị của các phép so sánh, giá trị đúng tương ứng với 1, sai tương ứng với 0

#include <stdio.h>

int main(){

printf("%d ", 100 > 50);

printf("%d ", 20 <= 30);

printf("%d ", 80 >= 100);

printf("%d ", 20 < 10);

printf("%d ", 10 != 20);

printf("%d ", 10 == 10);

return 0;

}

Output : 1 1 0 0 1 1

**4.Toán Tử Logic**

Có 3 toán tử logic là AND (&&) , OR (||) , NOT (!), các toán tử logic này được sử dụng để kết hợp nhiều toán hạng (biểu thức so sánh hoặc giá trị đúng sai) và sẽ trả về đúng hoặc sai.

Phần này bạn cần xem lại cách hoạt động của 3 cổng logic trên nếu chưa nắm vững.

**Chú ý**: Trong C thì tất cả các giá trị khác 0 được coi là giá trị đúng.

|  |  |  |  |
| --- | --- | --- | --- |
| **Toán tử** | **Ý nghĩa** | **Cách hoạt động** | **Ví dụ** |
| && | Toán tử AND (Và) | Cho giá trị đúng khi mọi toán hạng có giá trị đúng, sai trong các trường hợp còn lại | (10 < 20) && (20 == 20) : Đúng |
| || | Toán tử OR (Hoặc) | Cho giá trị sai khi mọi toán hạng có giá trị sai, đúng trong các trường hợp còn lại | (10 > 30) && (30 == 30) : Đúng |
| ! | Toán tử NOT (Phủ định) | Phủ định của đúng là sai, phủ định của sai là đúng | !(20 == 20) : Sai |

Ví dụ : Bạn có thể sử dụng code để in ra giá trị của các biểu thức

#include <stdio.h>

int main(){

//true && true = true

int res1 = (10 < 20) && (20 >= 20);

//true && true && false = false

int res2 = (10 < 20) && (20 == 20) && (5 > 10);

//false || false || true = true

int res3 = (10 > 20) || (20 < 10) || (5 == 5);

// !(true) = false

int res4 = !(10 < 20);

//!(true && true) = !(true) = false

int res5 = !((20 < 30) && (30 > 10));

printf("%d %d %d %d %d\n", res1, res2, res3, res4, res5);

return 0;

}

Output : 1 0 1 0 0

**5. Toán Tử Tăng Giảm**

Để tăng giảm giá trị của biến lên 1 đơn vị bạn có thể sử dụng toán từ ++, hoặc giảm giá trị của biến đi 1 đơn vị bằng toán tử -- .

* ++a : Tăng trước
* a++ : Tăng sau
* --a : Giảm trước
* a-- : Giảm sau

Ví dụ 1:

#include <stdio.h>

int main(){

int n = 100;

++n; // n = 101

printf("%d ", n);

n++; // n = 102

printf("%d ", n);

n--; // n = 101

printf("%d ", n);

--n; // 100

printf("%d", n);

return 0;

}

Output : 101 102 101 100

Ví dụ 2 : Sự khác nhau giữa n++ và ++n, tương tư như n-- và --n

n++ : Giá trị của n sẽ được sử dụng, sau đó n mới được tăng lên 1 đơn vị

++n : Giá trị của n sẽ được tăng lên luôn, sau đó sử dụng giá trị mới của n

#include <stdio.h>

int main(){

int n = 100;

printf("%d ", n++); // In 100 sau đó tăng n lên 101

printf("%d ", n); // 101

printf("%d ", ++n); // tăng n lên 102 sau đó in 102

printf("%d", n); // 102

return 0;

}

Output : 100 101 102 102

**6. Toán Tử Điều Kiện (3 Ngôi)**

Cú pháp : **[Biểu thức so sánh] ? [Giá trị trả về 1] : [Giá trị trả về 2];**

Ý nghĩa : Phần thứ nhất trong toán tử này sẽ có giá trị đúng hoặc sai, nếu phần này có giá trị đúng thì biểu thức sẽ trả về giá trị 1, ngược lại nếu phần này có giá trị sai thì biểu thức sẽ trả về giá trị 2

Ví dụ : int n = 10 < 20 ? 100 : 200; thì n sẽ có giá trị là 100, vì 10 < 20 có giá trị là đúng nên n được gán giá trị là 100 thay vì 200

#include <stdio.h>

int main(){

int n = 10 < 20 ? 28 : 82;

printf("%d\n", n);

n = (50 < 50) && (10 > 3) ? 28 : 82;

printf("%d\n", n);

return 0;

}