6.5.5 Multiplicative operators

Syntax

1 *multiplicative-expression:*
cast-expression
multiplicative-expression ***** *cast-expression*
multiplicative-expression **/** *cast-expression*
multiplicative-expression **%** *cast-expression*

Constraints

2 Each of the operands shall have arithmetic type. The operands of the **%** operator shall have integer type.

Semantics

3 The usual arithmetic conversions are performed on the operands.

4 The result of the binary ***** operator is the product of the operands.

5 The result of the **/** operator is the quotient from the division of the first operand by the second; the result of the **%** operator is the remainder. In both operations, if the value of the second operand is zero, the behavior is undefined.

6 When integers are divided, the result of the **/** operator is the algebraic quotient with any fractional part discarded.[90] If the quotient **a/b** is representable, the expression **(a/b)*b + a%b** shall equal **a**.

6.5.6 Additive operators

Syntax

1 *additive-expression:*
multiplicative-expression
additive-expression **+** *multiplicative-expression*
additive-expression **-** *multiplicative-expression*

Constraints

2 For addition, either both operands shall have arithmetic type, or one operand shall be a pointer to an object type and the other shall have integer type. (Incrementing is equivalent to adding 1.)

3 For subtraction, one of the following shall hold:

— both operands have arithmetic type;

---

90) This is often called ‘‘truncation toward zero’’.