**Operators in C**

**Objectives**

* Use arithmetic operators in C programs.
* Correctly apply the precedence of arithmetic operators
* Use relational operators in C programs.

**Arithmetic Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | Arithmetic Operator | C Expression | Example |
| Addition | + | no1 + 8 | 5 + 6 = 11 |
| Subtraction | - | value – no2 | 7 - 2 = 5 |
| Multiplication | \* | qty \* price | 4 \* 10.5 = 42.0 |
| Division | / | tot / 3 | 100 / 3 = 33 |
| Remainder | % | no1 % no2 | 10 % 3 = 1 |

Arithmetic operators are binary operators.

**Operator Precedence and Associativity**

* Operator precedence establishes the priority of an operator in relationship to all other operators.
* Parentheses can be used to modify the normal order of execution of an expression.
* Operator associativity establishes the order in which operators of the same precedence are to be executed.

**Operator Precedence of Arithmetic Operators**

|  |  |  |
| --- | --- | --- |
| **Order** | Operator(s) | Associativity |
| 1 | () Parentheses | Left to right |
| 2 | \* Multiplication  / Division  % Remainder | Left to right |
| 3 | + Addition  - Subtraction | Left to right |

**Equality and Relational Operators**

Equality and relational operators test the relationship between two expressions and yields true or false

|  |  |  |
| --- | --- | --- |
| **Operation** | Operator | C Expression |
| equal to | == | x == y |
| not equal to | != | x != y |
| greater than | > | x > 30 |
| less than | < | x < y |
| greater than or equal to | >= | x >= ( 6 + y ) |
| less than or equal to | <= | x <= y |

**Operator Precedence Revisited**

|  |  |  |
| --- | --- | --- |
| **Order** | Operator(s) | Associativit y |
| 1 | () | Left to right |
| 2 | ! | Right to left |
| 3 | \* / % | Left to right |
| 4 | + - | Left to right |
| 5 | < <= > >= | Left to right |
| 6 | == != | Left to right |
| 7 | = | Right to left |

**Logical Operators**

Used to form more complex conditions by combining simple conditions.

|  |  |  |
| --- | --- | --- |
| **Operation** | Operator | C Expression |
| Logical AND | && | gender = 1 && age >= 65 |
| Logical OR | || | semesterAverage >= 90 || finalExam >= 90 |
| Logical NOT | ! | ! ( grade == ‘F’) |

**Cast operator**

* Cast operators force the conversation of a value to a specified type. It is called explicit conversion.
* It is formed by placing parentheses around a data type name.

Example

int total = 203;

int count= 5;

float average;

average = ( float ) total / count;

**Converting between types implicitly**

* Arithmetic expressions can be evaluated only in which the operands’ data types are identical. To ensure this, the compiler performs an operation called implicit conversion on selected operands.

Example: In an expression containing the data types int and float, copies of *int* operands are made and promoted to *float.*

**Assignment Operators**

* There are several assignment operators for abbreviating assignment expressions.

*variable* = *variable* operator *expression*;

can be written as

*variable* operator= *expression;*

where operator is one of the binary operators +, -, \*, / or %

**Increment and Decrement operators**

++ increment operator

-- decrement operator

k = ++n; // prefix increment:

k = n++; // postfix increment:

k = --n; // prefix decrement:

k = n--; // postfix decrement:

**Selection**

* In solving a problem we can make different choices depending on certain conditions - i.e. we make decisions
* The same can be done in programming as a part of decision making
* Note that even though selection is a separate construct to sequence, the two are combined in the overall solution, and remember that one doesn’t replace the other
* There may be alternative steps that could be taken subject to a particular condition

