ICT @ Mzuni

Introduction to programming with C

By: Chimango Nyasulu, PhD

ICT @ Mzuni

Outline

- Operators
- Operator precedence and associativity
- Order of evaluation
- Functions

Operators in C

In C programming, operators are special symbols that perform operations on variables and values.

- Understanding operators, their precedence, associativity, and how they interact with functions is crucial for effective programming.

Operators in C can be classified into several categories:

1. **Arithmetic Operators**: Used for mathematical calculations.
   + (Addition)
   - (Subtraction)
   * (Multiplication)
   / (Division)
   % (Modulus)

ICT @ Mzuni

Operators in C

2. **Relational Operators**: Used to compare two values.

   == (Equal to)
   != (Not equal to)
   > (Greater than)
   < (Less than)
   >= (Greater than or equal to)
   <= (Less than or equal to)

- Return **1** if True and **0** if False

ICT @ Mzuni

Operators in C

3. **Logical Operators** are used to perform logical operations on boolean (true/false) values. The result of these operations is also a boolean value.

The three primary logical operators in C are:

1. **Logical AND (&&):** This operator returns true if both operands are true. If either operand is false, it returns false.

```
int a = 5, b = 10;
if (a > 0 && b > 0) {
    // This block will execute because both conditions are true
}
```

ICT @ Mzuni

Operators in C

The three primary logical operators in C are:

2. **Logical OR (||)**: This operator returns true if at least one of the operands is true. It only returns false if both operands are false.

```
int a = 5, b = -10;
if (a > 0 || b > 0) {
    // This block will execute because at least one condition is true
}
```

Operators in C

The three primary logical operators in C are:

3. **Logical NOT (!)**: This operator negates the boolean value of its operand. If the operand is true, it returns false, and vice versa.

```
int a = 5;
if (!(a < 0)) {
    // This block will execute because !(false) is true
}
```

ICT @ Mzuni

Operators in C

4. **Assignment Operators**: are used to assign values to variables.

A. **Basic Assignment Operator**: = Assigns the value on the right to the variable on the left.

```
int a = 5; // Assigns 5 to a
```

B. **Compound Assignment Operators**: These operators perform an arithmetic operation and assignment in one step.

```
a += 3; // Equivalent to a = a + 3; (a becomes 8)
```

```
a -= 2; // Equivalent to a = a - 2; (a becomes 6)
```

```
a *= 2; // Equivalent to a = a * 2; (a becomes 12)
```

```
a /= 3; // Equivalent to a = a / 3; (a becomes 4)
```

Operators in C

5. **Increment and Decrement Operators**: are unary operators that are used to increase or decrease the value of a variable by one, respectively. It can be used in two forms:

A. **Postfix Increment (variable++)**: The value of the variable is returned first, and then it is incremented.

```
int a = 5;
int b = a++; // b gets the value 5, a becomes 6
```

B. **Prefix Increment (++variable)**: The variable is incremented first, and then the new value is returned.

```
int a = 5;
int b = ++a; // a becomes 6, b gets the value 6
```

Operators in C

C. **Postfix Decrement (variable--)**: The value of the variable is returned first, and then it is decremented.

```
int a = 5;
int b = a--; // b gets the value 5, a becomes 4
```

D. **Prefix Decrement (--variable)**: The variable is decremented first, and then the new value is returned.

```
int a = 5;
int b = --a; // a becomes 4, b gets the value 4
```

ICT @ Mzuni

Operators in C

6. Conditional Operator: is a ternary operator that provides a shorthand way of performing an if-else statement. It is represented by the ? and : symbols.

```
condition ? expression_if_true : expression_if_false;
```

7. Comma Operator (,): is a binary operator that evaluates its first operand, discards the result, and then evaluates and returns the second operand.

```
expression1, expression2
```

Operators in C

8. sizeof Operator: is used to determine the size (in bytes) of a data type, variable, or object.

- It is a compile-time operator, which means that it evaluates its operand and returns the size during the compilation process, not at runtime.

1. **For a data type:**

```
sizeof(type)
```

2. **For a variable:**

```
sizeof(variable)
```

Operators in C

9. Pointer Operators: These are used to manipulate pointers, which are variables that store memory addresses.

There are two main pointer operators:

1. **Address-of Operator (&)**: This operator is used to obtain the address of a variable. When you prefix a variable with &, it returns the memory address of that variable.

```
int x = 10;
int *ptr = &x; // ptr now holds the address of x
```

ICT @ Mzuni

Operators in C

There are two main pointer operators:

2. **Dereference Operator (*)**: This operator is used to access the value at the address that a pointer is pointing to. When you prefix a pointer with *, it dereferences the pointer and allows you to manipulate the value at that address.

```
int y = *ptr; // y now holds the value of x (which is 10)
*ptr = 20;    // now x is changed to 20 through the pointer
```

ICT @ Mzuni

Operator Precedence and Associativity

- **Operator precedence** determines the order in which operators are evaluated in expressions. Operators with higher precedence are evaluated before those with lower precedence.

- **Associativity** determines the order of evaluation for operators of the same precedence level. It can be left-to-right or right-to-left.

```
int a = 5, b = 10, c;
c = a + b * 2;  // b * 2 is evaluated first because * has higher precedence than +
// c = 5 + 20 = 25
```

ICT @ Mzuni

Functions

- Functions in C are blocks of code that perform a specific task and can be reused throughout a program. They can take parameters and return a value.

Function Definition:

```
return_type function_name(parameter_list) {
    // Function body
    return value; // Optional return statement
}
```