ICT @ Mzuni

Introduction to programming with C

By: Chimango Nyasulu, PhD

ICT @ Mzuni

MZUZU UNIVERSITY

Outline

- Variable definition
- Variable declarations
- Storage classes and scope of a variable
- Variable initialization
- Operators by category
- Operator precedence and associativity
- Order of evaluation
- Functions

ICT @ Mzuni

Variable Definition

A variable definition allocates storage for the variable and specifies its type.

- When you define a variable, you essentially tell the compiler to reserve space in memory for it.

```
int age; // This line defines a variable named 'age' of type 'int'
```

ICT @ Mzuni

Variable Declarations

A variable declaration introduces a variable to the program but does not allocate space for it if it is already defined elsewhere.

- It tells the compiler about the type of the variable and its name.

```
extern int age; // This line declares a variable named 'age' which is defined elsewhere
```

ICT @ Mzuni

Variable Initialization

Variable initialization is the process of assigning a value to a variable at the time of its definition.

- This is important to avoid undefined behavior from using uninitialized variables.

```
int age = 25; // This line defines and initializes 'age' with the value 25
```

ICT @ Mzuni

Storage Classes

The storage class of a variable determines its lifetime, visibility, and scope. C provides several storage classes:

1. **Automatic Storage Class (auto):**
    - Default for local variables.
    - Lifetime: exists until the block in which it is defined exits.
    - Scope: limited to the block.

```
void function() {
    int x; // 'x' is automatic; it is created when the function is called and destroyed when it returns.
}
```

ICT @ Mzuni

Storage Classes

2. Static Storage Class (static):

- Variables retain their value between function calls.
- Lifetime: exists for the duration of the program.
- Scope: limited to the block in which it is defined or the file if defined at file scope.

```
void function() {
    static int count = 0; // Retains its value between calls to function()
    count++;
}
```

ICT @ Mzuni

Storage Classes

3. **External Storage Class (extern):**

- Used for variables that are defined outside of the current file or function.
- Lifetime: exists for the duration of the program.
- Scope: global, accessible from any file within the same program.

```
int globalVar; // Definition
void function() {
    extern int globalVar; // Declaration
}
```

ICT @ Mzuni

Storage Classes

4. **Register Storage Class (register):**

- Suggests to the compiler to store the variable in a CPU register for faster access.
- Lifetime: exists until the block in which it is defined exits.
- Scope: limited to the block.

```
void function() {
    register int speed; // Requests to store 'speed' in a register
}
```

ICT @ Mzuni

Scope of a Variable

The scope of a variable refers to the region of the program where the variable is accessible.

1. **Local Scope**: Variables defined within a function or block are only accessible within that function/block.

```
void function() {
    int localVar = 5; // Only accessible within 'function'
}
```

2. **Global Scope**: Variables defined outside of all functions are accessible from anywhere in the file or any other file that declares them.

```
int globalVar = 10; // Accessible from all functions in this file
```