ICT @ Mzuni

Introduction to programming with C

By: Chimango Nyasulu, PhD

ICT @ Mzuni

MZUZU UNIVERSITY

Outline

- Characteristics of C
- Compilers and Interpreters
- Lexical conventions
- # include files
- Keywords
- Constants
- Basic Data Types
- Modifying data types

ICT @ Mzuni

Introduction to C

- C is a high-level programming language that was developed in the early 1970s at Bell Labs by Dennis Ritchie.
- It has since become one of the most widely used programming languages in the world, particularly for system programming, embedded systems, and application development.
- C is known for its efficiency, flexibility, and ability to interact closely with hardware.

ICT @ Mzuni

Characteristics of C

1. **Efficiency:** C is a compiled language, which means that programs written in C are translated into machine code, resulting in fast execution times.

2. **Low-Level Access:** C provides low-level access to memory through the use of pointers, allowing developers to manipulate hardware directly.

3. **Portability:** C programs can be compiled and run on different types of computer systems with little or no modification, making it a portable language.

4. **Rich Library Support:** C has a standard library that provides numerous built-in functions for various tasks.

    - E.g. mathematical computations, string manipulations, and data handling.

ICT @ Mzuni

Characteristics of C

5. **Structured Language:** C encourages structured programming, allowing for better organization and readability of code through functions, control structures, and data types.

6. **Modularity:** C supports modular programming, enabling developers to break down programs into smaller, manageable functions and modules.

7. **Flexibility:** C allows the use of both high-level and low-level programming techniques, giving developers flexibility in their approach to problem-solving.

8. **Static Typing:** C is statically typed, meaning that variable types are known at compile time, which can help catch errors early in the development process.

ICT @ Mzuni

Compilers and Interpreters

- **Compiler:** A compiler is a program that translates source code written in a high-level programming language (like C) into machine code or intermediate code.
  - The resulting executable can be run directly on the hardware.
  - C is typically compiled, which means the entire program is translated before execution.
  - E.g. GCC (GNU Compiler Collection), Clang, and MSVC (Microsoft Visual C++).
- **Interpreter:** An interpreter translates high-level code into machine code line by line at runtime.

ICT @ Mzuni

Lexical Conventions

Lexical conventions in C define the rules for how the source code is structured and interpreted.

Here are some important aspects:

1. **Tokens:** The smallest elements in a C program are tokens (keywords, identifiers, constants, string literals, operators, and punctuation marks).
    - Keywords: Reserved words that have a special meaning (e.g., int, return, if, else).
    - Identifiers: Names given to variables, functions, and arrays. Identifiers must begin with a letter or underscore and can contain letters, numbers, and underscores.
    - Constants: Fixed values like numbers (e.g., 42, 3.14) or string literals (e.g., "Hello, World!").

Lexical Conventions

Here are some important aspects:

2. **Comments:** C supports single-line comments (preceded by //) and multi-line comments (enclosed between /* and */).

3. **Whitespace:** Spaces, tabs, and newlines are ignored by the compiler, except where they are necessary to separate tokens.

4. **Preprocessor Directives:** Lines beginning with # (like #include, #define) are preprocessor directives.

    - They are processed before compilation and are used for including libraries and defining constants.

ICT @ Mzuni

Lexical Conventions

Here are some important aspects:

5. **Semicolons:** Statements in C are typically terminated with a semicolon (;), indicating the end of a statement.

6. **Braces:** C uses curly braces {} to define the beginning and end of code blocks, such as those for functions or control structures like if, for, and while.

Structure of a C Program

A basic C program consists of several key components that follow a specific structure.

1. **Preprocessor Directives:** These are commands that give instructions to the compiler to preprocess the source code before actual compilation (#include for including libraries).

2. **Function Declarations**: Any functions that will be used in the program can be declared here.

3. **The main Function:** This is the entry point of every C program. The execution of the program starts from this function.

4. **Variable Declarations:** Variables are declared here.

5. **Executable Statements:** This includes the statements that define the program's logic.

6. **Return Statement:** This indicates the end of the main function and returns a value to the operating system.

Example of a Basic C Program

```
#include <stdio.h>        // Preprocessor directive to include standard input-output library

int main()
{ // Main function where execution begins

   // Executable statements
   printf("I love Programming");
   printf("\n");

   return 0;     // Return statement indicating successful execution
}
```

ICT @ Mzuni

Include Files

Include files in C are used to incorporate functionality from external libraries or parts of the standard library into your program.

Common include files include:

<stdio.h>: Standard Input/Output functions (e.g., printf, scanf).
<stdlib.h>: General utilities (e.g., memory allocation, random numbers).
<string.h>: String handling functions (e.g., strlen, strcpy).
<math.h>: Mathematical functions (e.g., sin, cos, sqrt).
<ctype.h>: Character handling functions (e.g., isdigit, isalpha).

```
#include <stdio.h>
#include <stdlib.h>
```

ICT @ Mzuni

Keywords

Keywords are reserved words in C that have special meaning and cannot be used as identifiers (variable names).

Some important keywords include:

int: Defines integer type.
float: Defines single-precision floating-point type.
double: Defines double-precision floating-point type.
char: Defines character type.
if, else, for, while, return: Control flow statements.
void: Indicates no value.

```
#define PI 3.14159
const int MAX_USERS = 100;
```

ICT @ Mzuni

Basic Data Types

C provides several basic data types:

int: Represents integer values. Typically, it can be 16, 32, or 64 bits depending on the system.
float: Represents single-precision floating-point numbers (usually 32 bits).
double: Represents double-precision floating-point numbers (usually 64 bits).
char: Represents single characters (typically 8 bits).

```
int age = 30;
float height = 5.9;
double pi = 3.14159;
char initial = 'A';
```

ICT @ Mzuni

More on Floating Points

Floating-point types (float and double) are used to represent real numbers. They can express a wide range of values, but they have precision limitations:

float: Typically has 7 decimal digits of precision.
double: Typically has 15 decimal digits of precision.

It is important to be cautious of precision errors when performing arithmetic with floating-point numbers.

```
float a = 1.0 / 3.0;  // May not be exactly 0.33333
double b = 1.0 / 3.0; // More precise than float
```

ICT @ Mzuni

Modifying Data Types

C allows you to modify basic data types using the following modifiers:

long: Increases the size of an integer or floating-point type (e.g., long int, long double).
short: Decreases the size of an integer type (e.g., short int).
signed: Indicates that the variable can hold both negative and positive values (default for int).
unsigned: Indicates that the variable can hold only non-negative values.

```
long int big_number = 100000L;
short int small_number = 10;
unsigned int positive_number = 25;
```