

introduction to computer programming structured cobol

Introduction to Computer Programming Structured COBOL

Computer programming is a fundamental skill in the modern world, enabling individuals to create applications, manage data, and solve complex problems through code. One of the programming languages that has stood the test of time is COBOL, which stands for Common Business-Oriented Language. Developed in the late 1950s, COBOL was designed specifically for business applications and data processing. This article will provide an introduction to structured COBOL, exploring its syntax, features, advantages, and the role it continues to play in the computing landscape.

What is COBOL?

COBOL is a high-level programming language that was developed to meet the needs of business, finance, and administrative systems for companies and governments. Its design focuses on readability, making it easier for non-programmers to understand the code. One of the defining characteristics of COBOL is its English-like syntax, which includes long variable names and descriptive keywords.

Historical Background

The inception of COBOL can be traced back to a conference in 1959, where representatives from various businesses and government agencies gathered to create a standardized programming language. The first specification was released in 1960, and COBOL quickly gained popularity for its ability to handle large amounts of data and perform batch processing tasks.

Throughout its history, COBOL has undergone several revisions, with COBOL 85 being one of the most significant updates. This version introduced structured programming concepts, which allowed for better organization and readability of code. The latest standard, COBOL 2002, added support for object-oriented programming, further enhancing its functionality.

Structured Programming in COBOL

Structured programming is a programming paradigm that promotes clear, logical structure in code. It encourages the use of control structures like loops and conditionals, which help reduce complexity and improve maintainability. Structured COBOL adheres to these principles and allows programmers to write code that is modular and easy to understand.

Key Features of Structured COBOL

1. Divisions: COBOL programs are divided into four main divisions:

- Identification Division: Contains metadata about the program, including its name and author.
- Environment Division: Specifies the environment in which the program will run, including input and output devices.
- Data Division: Defines the variables and data structures used in the program.
- Procedure Division: Contains the actual logic of the program, where the computations and processing occur.

2. Sections and Paragraphs: Within the Procedure Division, code can be organized into sections and paragraphs, allowing for better structure and readability.

3. Control Structures: Structured COBOL supports various control structures, including:

- IF Statements: Used for conditional execution of code.
- PERFORM Statements: Used to invoke sections or paragraphs, promoting code reuse.
- EVALUATE Statements: Similar to switch-case statements in other languages, allowing for multi-way

branching.

4. Structured Data Types: COBOL provides a rich set of data types, including:

- Numeric: For handling numbers.
- Alphabetic: For handling characters.
- Alphanumeric: For handling strings of characters.

5. File Handling: COBOL has robust file handling capabilities, allowing for the processing of sequential and indexed files, which is essential for business applications.

Advantages of Using Structured COBOL

Despite being over six decades old, COBOL remains relevant in today's programming landscape. Here are some of its key advantages:

1. Readability: The English-like syntax makes COBOL code easy to read and understand, which is particularly useful in business environments where stakeholders may not have a technical background.
2. Business Focus: COBOL is designed specifically for business applications, making it well-suited for tasks like payroll processing, banking transactions, and inventory management.
3. Legacy Systems: Many large organizations still rely on COBOL for their legacy systems. Learning COBOL can be advantageous for developers working in industries such as finance and government, where these systems are prevalent.
4. Data Processing: COBOL is optimized for data processing tasks, making it an excellent choice for applications that require handling large volumes of data.
5. Community and Resources: With its long history, COBOL has a vast array of resources, documentation, and community support available for developers.

Getting Started with Structured COBOL

For those interested in learning structured COBOL, there are several steps to follow:

1. Setting Up the Environment

To start programming in COBOL, you'll need a development environment. Here are some options:

- Compiler: Install a COBOL compiler like GnuCOBOL or Micro Focus Visual COBOL. These compilers will allow you to write and execute COBOL code.
- IDE: Integrated Development Environments (IDEs) like Eclipse with COBOL plugins or Visual Studio Code can enhance your coding experience, providing features like syntax highlighting and debugging tools.

2. Learning the Basics

Before diving into complex projects, familiarize yourself with the syntax and structure of COBOL. Here are some essential concepts to understand:

- Data Types: Learn about the different data types available in COBOL and how to declare variables.
- Control Structures: Practice using IF, PERFORM, and EVALUATE statements to control the flow of your programs.
- File Handling: Understand how to read from and write to files, as this is crucial for data processing tasks.

3. Writing Your First COBOL Program

Here is a simple example of a COBOL program that calculates the sum of two numbers:

```
``cobol  
IDENTIFICATION DIVISION.  
PROGRAM-ID. SumCalculator.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01 Num1 PIC 9(5).  
01 Num2 PIC 9(5).  
01 Sum PIC 9(5).  
PROCEDURE DIVISION.  
DISPLAY 'Enter first number: '.  
ACCEPT Num1.  
DISPLAY 'Enter second number: '.  
ACCEPT Num2.  
ADD Num1 TO Num2 GIVING Sum.  
DISPLAY 'The sum is: ' Sum.  
STOP RUN.  
``
```

This program showcases basic input and output operations, variable declarations, and arithmetic operations in COBOL.

Conclusion

Structured COBOL continues to be a vital programming language in the business world, offering

unique advantages for data processing and enterprise applications. Its readability, focus on business needs, and robust features make it a preferred choice for many organizations. By understanding the principles of structured programming and getting hands-on experience with COBOL, aspiring programmers can build a solid foundation in this enduring language. Whether you are maintaining legacy systems or developing new applications, COBOL remains a valuable skill in the ever-evolving landscape of computer programming.

Frequently Asked Questions

What is COBOL and why is it important in programming?

COBOL, which stands for Common Business-Oriented Language, is a high-level programming language designed for business applications. It is important because it has been widely used in enterprise environments for decades, particularly in financial and administrative systems.

What are the basic structure and syntax rules in COBOL?

COBOL programs are divided into four divisions: Identification, Environment, Data, and Procedure. Each division has a specific purpose, and the syntax follows a strict format with a focus on readability, using English-like statements.

How does COBOL handle data types and structures?

COBOL uses a variety of data types, such as numeric, alphabetic, and alphanumeric, and supports complex data structures through the use of records and arrays. Data definitions are specified in the Data Division, allowing for structured data management.

What are the key differences between structured programming and COBOL?

Structured programming is a programming paradigm that emphasizes dividing programs into smaller,

manageable sections. While COBOL supports structured programming concepts, it is traditionally procedural and may include more verbose syntax compared to modern structured languages.

Can COBOL be used for modern applications, and if so, how?

Yes, COBOL can be used for modern applications, especially in legacy systems. It can be integrated with newer technologies through web services, APIs, and can also be run on cloud platforms, enabling COBOL applications to interact with contemporary systems.

What role does COBOL play in the context of legacy systems?

COBOL plays a critical role in legacy systems, especially in industries like banking and insurance, where many core applications are written in COBOL. Its stability and reliability make it essential for maintaining and updating these systems.

What resources are available for learning COBOL programming?

There are numerous resources available for learning COBOL, including online courses, textbooks, and tutorials. Additionally, organizations like IBM and Micro Focus provide COBOL development tools and documentation to assist learners.

[Introduction To Computer Programming Structured Cobol](#)

Introduction To Computer Programming Structured Cobol

Related Articles

- [integumentary system worksheet 1](#)
- [intercultural communication encounters](#)
- [is shift technologies going out of business](#)

[Back to Home](#)