Go Programming Language

Introduction

Department of Computer Science and Engineering

Go Programming Language

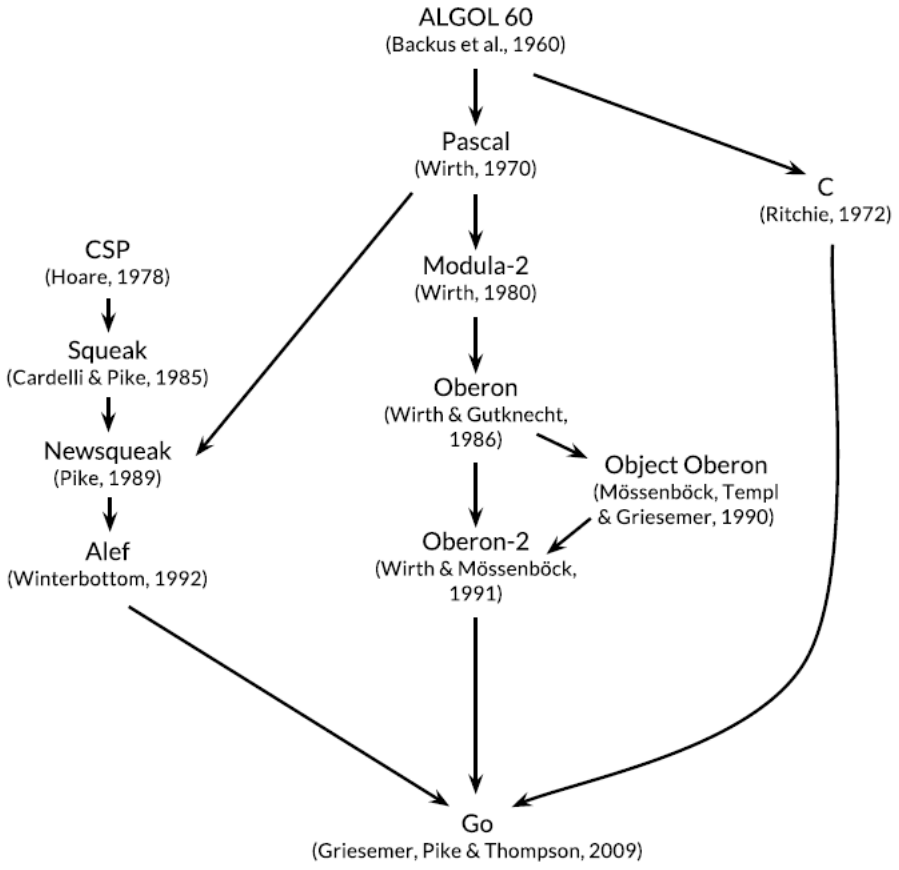
Introduction

- Go is an open source programming language that makes it easy to build simple, reliable, and efficient software
- Go was conceived in September 2007 by Robert Griesemer, Rob Pike, and Ken Thompson, while working at Google, and was announced in November 2009.
- Go is especially well suited for building infrastructure like networked servers, and tools and systems for programmers
- It has become popular as a replacement for untyped scripting languages because it balances expressiveness with safety.
- Untyped languages, also known as dynamically typed languages, are programming languages that do not make you define the type of a variable

Go Programming Language

Genesis

- Go inherited its expression syntax, control-flow statements, basic data types, call-by-value parameter passing, pointers, from C.
- In Communicating Sequential Processing, a program is a parallel composition of processes that have no shared state; the processes communicate and synchronize using channels.
- Go inherited the fundamental concepts of concurrency from CSP.
- Oberon-2 influenced the syntax for packages, imports, and declarations, and Object Oberon provided the syntax for method declarations



Go Programming Language

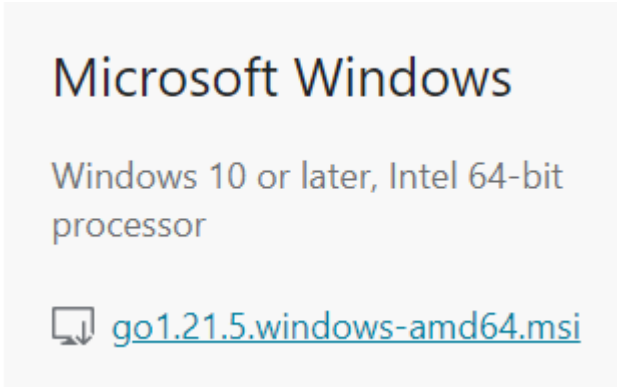
Introduction

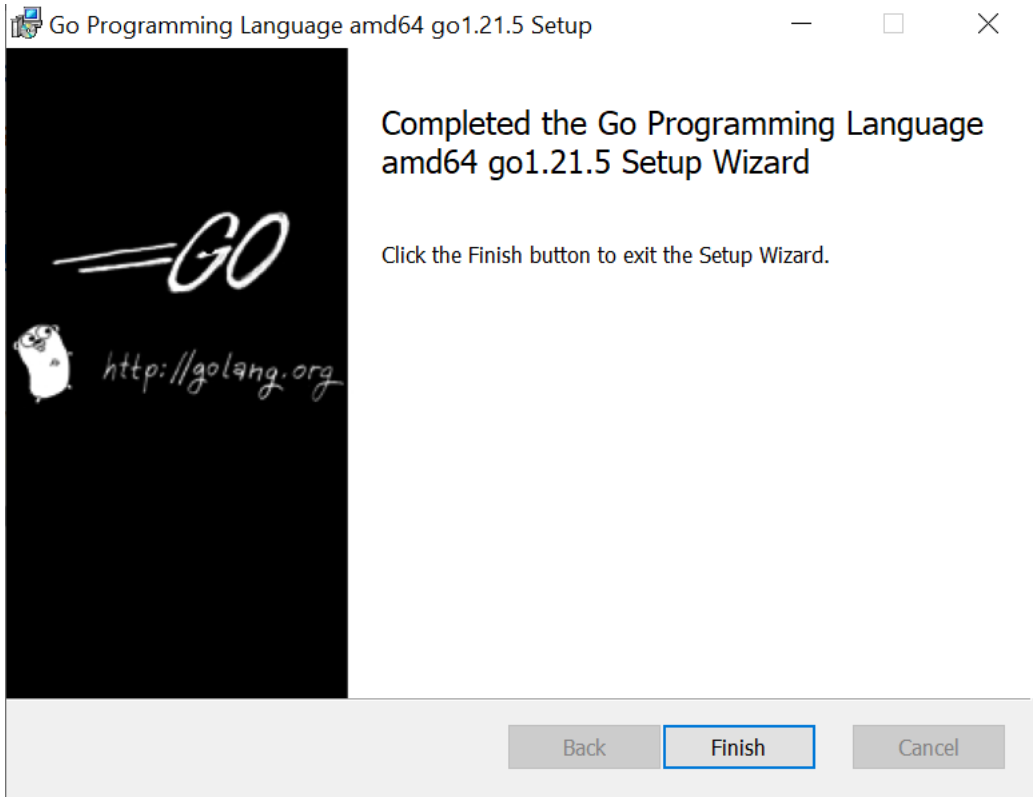
- Go has no implicit numeric conversions, no constructors or destructors, no operator overloading, no default parameter values, no inheritance, no generics, no exceptions, no macros, no function annotations, and no thread-local storage
- Go's built-in data types and most library data structures are crafted to work naturally without explicit initialization or implicit constructors, so relatively few memory allocations and memory writes are hidden in the code.
- Go's aggregate types (structs and arrays) hold their elements directly, requiring less storage and fewer allocations and pointer indirections than languages that use indirect fields
- Go's standard library, provides clean building blocks and APIs for I/O, text processing, graphics, cryptography, networking, and distributed applications, with support for many standard file formats and protocols.

Go Programming Language

Installation

- https://go.dev/doc/install





Go Programming Language

First program

```
C:\PESIT\Go>more helloworld.go
package main

import "fmt"

func main() {
   fmt.Println("Hello Students!!!")
}

C:\PESIT\Go>go run helloworld.go
Hello Students!!!
```

```
C:\PESIT\Go>go build helloworld.go

C:\PESIT\Go>dir
 Volume in drive C is OS
 Volume Serial Number is 40E2-F49F

 Directory of C:\PESIT\Go

09-01-2024  16:00    <DIR>          .
09-01-2024  16:00    <DIR>          ..
14-12-2023  09:26         2,904,832 gobook.pdf
09-01-2024  16:00         1,897,472 helloworld.exe
09-01-2024  15:55                85 HelloWorld.go
14-12-2023  11:01         2,897,615 Introducing Go -
20-12-2023  09:39           323,072 Syllabus.doc
14-12-2023  11:01         5,307,238 The Go Programmin
09-01-2024  15:59         1,654,342 Unit 1 - 01 Intro
               7 File(s)     14,984,656 bytes
               2 Dir(s)  49,369,362,432 bytes free
```

Go Programming Language

First program

- Go code is organized into packages, which are similar to libraries or modules in other languages.
- A package consists of one or more **.go** source files in a single directory that define what the package does.
- Each source file begins with a package declaration, here package **main**, that states which package the file belongs to, followed by a list of other packages that it imports, and then the declarations of the program that are stored in that file.
- **fmt** package contains functions for printing formatted output and scanning input.
- **Println** is one of the basic output functions in fmt; it prints one or more values, separated by spaces, with a newline character at the end so that the values appear as a single line of output.

Go Programming Language

First program

- Package **main** is special. It defines a standalone executable program, not a library.
- Within package main the function main is also special - it’s where execution of the program begins.
- Whatever main does is what the program does. Of course, main will normally call upon functions in other packages to do much of the work, such as the function fmt.Println.
- A program will not compile if there are missing imports or if there are unnecessary ones. This strict requirement prevents references to unused packages from accumulating as programs evolve.
- The import declarations must follow the package declaration. After that, a program consists of the declarations of functions, variables, constants, and types (introduced by the keywords func, var, const, and type)
- The order of declarations does not matter

Go Programming Language

First program

- A function declaration consists of the key word func, the name of the function, a parameter list (empty for main), a result list (also empty here), and the body of the function—the statements that define what it does—enclosed in braces.
- Go does not require semicolons at the ends of statements or declarations, except where two or more appear on the same line.
- In effect, newlines following certain tokens are converted into semicolons, so where newlines are placed matters to proper parsing of Go code
- The opening brace **{** of the function must be on the same line as the end of the **func** declaration, not on a line by itself

Go Programming Language

Command line arguments

- The **os** package provides functions and other values for dealing with the operating system in a platform-independent fashion.
- Command-line arguments are available to a program in a variable named Args that is part of the os package; thus its name anywhere outside the os package is os.Args.
- The variable os.Args is a slice of strings.
- Slices are a fundamental notion in Go. Think of a slice as a dynamically sized sequence s of array elements where individual elements can be accessed as s[i] and a contiguous subsequence as s[m:n].
- The first element of os.Args, os.Args[0], is the name of the command itself
- The other elements are the arguments that were presented to the program when it started execution
- A slice expression of the form s[m:n] yields a slice that refers to elements m through n - 1

Go Programming Language

Command line arguments

```
package main

import ("fmt"
        "os")

func main() {
   fmt.Println(os.Args[0])
   fmt.Println(os.Args[1])
   fmt.Println(os.Args[2])
   fmt.Println(os.Args[0:3])
   fmt.Println(os.Args[0:])
   fmt.Println(os.Args[:3])
}
```

```
PS C:\pesit\go> go run args.go one two
C:\Users\SURESH~1\AppData\Local\Temp\go-build2785150662\b001\exe\args.exe
one
two
[C:\Users\SURESH~1\AppData\Local\Temp\go-build2785150662\b001\exe\args.exe one two]
[C:\Users\SURESH~1\AppData\Local\Temp\go-build2785150662\b001\exe\args.exe one two]
[C:\Users\SURESH~1\AppData\Local\Temp\go-build2785150662\b001\exe\args.exe one two]
```

THANK YOU

Suresh Jamadagni

Department of Computer Science and Engineering