

שאלה 2:

למדו את שפת GO מתוך התיעוד הרשמי שלה:

- א. מהם בנאי הטיפוסים בשפה? האם יש שם בנאים שלא מופיעים בקורס? האם יש בנאים תיאורטיים שנלמדו בקורס ולא מופיעים בשפה? מיהם?
- ב. מהם הטיפוסים האטומיים?
- ג. נתח את שפת GO לפי הקריטריונים שנלמדו בקורס לסיווג מערכת טיפוסים.

(א)

Record Type : struct , hash table ,interface{}

Integral exponentiation : array , hash table ,

Union : pointer

Mapping : function,

ניתן להתייחס `Hash table, map` לשני הבנאים כי בפועל היא ממפה ערך `KEY` ל ערך שלו \ או טבלה של הערכים המתאימים לאותו המפתח.

the empty interface `interface{}` is an important base case because it can refer to an item of any concrete type. It is similar to the `Object` class in Java or C# and is satisfied by any type, including built-in types like `int`

Power set, Product.

(ב)

String, bool , complex64, complex128,
float32,float64,int16,int32,int64,int8,unit8,unit16,
unit32,unit64.

(ג)

1) קימת מערכת טיפוסים .

2) כמה מורכבת מערכת הטיפוסים (חזקה מאוד , יש לה מערכת טיפוסים , ניתן להגדיר טיפוס חדש , ניתן לעשות טיפוס עם מצביע לעצמו , וניתן לעשות הכול עם פונקציות ניתן לשמור מצביע לפונקציה בתוך משתנה וגם לשלוח מצביע של הפונקציה וגם להחזיר , ויש לה משהו דומה למנגנון ההורשה כפי שראינו בתיעוד (Go provides two features that replace class inheritance לכן היא מרמה 6).

3) מערכת טיפוסים היא אורתוגונלית, ניתן להפעיל כל בנאי טיפוסים על כל טיפוס .

For each type T and each non-negative integer constant n , there is an array type denoted $[n] T$

Pointers are available for all types

For a pair of types K, V , the type $\text{map}[K] V$ is the type of hash tables mapping type- K keys to type- V values

the empty interface `interface{}` is an important base case because it can refer to an item of any concrete type. It is similar to the `Object` class in Java or C# and is satisfied by any type, including built-in types like `int`

Function types are indicated by the `func` keyword; they take zero or more parameters and return zero or more values, all of which are typed

(4) השפה חזקה

ב Go אין casting אוטומטי של טיפוסים. חיבור של int ו float גורר שגיאת קומפילציה

(5)

Statically Typed like java

(6)

VAR (semi implicit typing) נשתמש במילה השמורה

והוא יסיק הטיפוס מעצמו. או x:=

אבל לא תמיד ניתן להסתמך על זה בדוגמה פלט של פונקציה

(7) שקילות טיפוסים (מבנית)

In formal language, Go's interface system provides [structural](#) rather than [nominal](#) typing.

(8) super flexibility