מושגים קטנים להסבר

Static/dynamic linking

Byte code

Static/dynamic type system

Dynamic binding

Strongly-typed/ weakly-typed variables

Garbage collection

Lazy evaluation

Pattern matching?

Reflection?

Memory access/page faults

היי

בתור מתכנתים יש לנו פעמים רבות את השאלה "איזו שפת תכנות טובה יותר" ?

JAVA נכון ?

רגע, אולי C ? C++ אולי ?

פייתון?

אני אחדד את השאלה

בתהליך של פיתוח תוכנה ישנן מס' "צמתים" בהם ניתן להחליט על שפת התכנות בה נרצה לממש את הפיצ'ר שלנו.

לעתים מדובר בפיצ'ר קטן שמשתלב בפרוייקט גדול, פיצ'ר קטן שהוא עצמאי לגמרי, ולעתים להתחיל פרוייקט / מוצר שלם מאפס. צריך לבחור באיזו שפה טוב לי, או עדיף לי, לעבוד. והבחירה קריטית.

אז בתור מתכנתים בצמתי הבחירה הנ"ל אנו נשאל "איזו שפת תכנות טובה לי יותר בפיצ'ר/פרוייקט X" ?

תשובה אחת – מהשרוול , "JAVA" וזהו. כאנשי מדע תשובות כאלה תמיד נשמעות לנו חשודות , נשענות על יותר מדי "אמונה".

תשובה מושכלת יותר שיותר נפוצה בשוק היא – לכל ייעוד/ סוג משימה יש שפה שמתאימה לו – ...

התשובה לשאלה ששאלתי היא מאוד מאוד סובייקטיבית :

1. אילו שפות אני מכיר
2. באילו שפות עובדים בפרוייקט/ בחברה בה אני עובד (לעתים נגזר מלמעלה)
3. דעות חיצוניות (הלך רוח), אופנות
4. איזו שפה יהיה לי "נוח" יותר לעבוד על הקוד
5. באיזו שפה יש זמינות גדולה יותר של כלים/ספריות שיסייעו לי (לעתים זה נגזר מתשתית הקיימת במערכת שעליה אני עובד – מערכת הפעלה, jboss)
6. איזו שפה תייצר קוד בטוח יותר ? קצר/קריא יותר? יעיל (מהיר) יותר ? חסכוני בזיכרון ?

(טובת הכלל)

יש ויכוחים עזים שניטשים בשאלות הללו

החיבור לנושא הסמינר "הגורם האנושי בהנדסת תוכנה" הוא לדעתי הדרך שבה אנו כבני אדם עוצרים בבולט מס' 3-4 ולא מסתמכים על נתונים אמפיריים. פעמים רבות איננו מודעים בכלל לתהליך הבחירה. סביר שנישאר באותן שפות שאנחנו מכירים/נוח לנו. בין צמתי הבחירה לרוב פשוט נמשיך.

תמיד מדובר בהשוואות - זה מול זה, הדבר שאני רגיל אליו והולך אליו באופן טבעי, לעומת הדבר שאיני מכיר ונמצא מעבר לפינה , ואולי דווקא הוא יעזור לי בצורה הטובה ביותר. כמובן שישנן חלופות רבות בשוק.

המחקר שאני מציג בא לסלק חלק מהגורמים הסובייקטיביים לבחירות אלא, ולהשוות בין מס' שפות תכנות לפי פרמטרים שונים .

הדרך שבה בוצע המחקר היא דרך מאגר קוד פתוח גדול בשם "Rosetta Code"

הסבר על המאגר

הסבר על פרדיגמות

הסבר על שיטת המחקר

דוגמה – לקרוא קובץ אבל לפני לעשות לו unzip . נרצה לפצל בין C לבין סקריפט.

אם הפיצ'ר קטן מאוד, סביר שלא נבחר לפצל אותו (אם התחלתי לכתוב quicksort בC והלכתי הביתה, למחרת מה יקרה? סביר שאסיים אותו בC, או לחלופין אזנח לחלוטין את המימוש ואלך לממש בשפה אחרת).

תוצאות +מושגים בדרך

הימורים בפורמט :

- מי מהפרדיגמות גרוע ביותר , טוב ביותר.

- ווינר הגדול, מפסיד הגדול.

- פי כמה.

תמציתיות

נבדקו הקבוצה הראשונה של פתרונות (בשתי בדיקות – אם עברו קומפילציה או לא – אותן תוצאות).

נמדדו ללא ריקים וללא הערות. אם למשימה מסוימת היו מס' פתרונות בשפה , נבחר הקצר מביניהם.

תוצאות

FUNC וSCR – נמצאו כתמציתיות ביותר

PROC וOO – משמעותית יותר שורות קוד.

לדוגמה תוכנית בJAVA תיקח 2.2-2.9 שורות קוד מאותה תוכנית בשפת FUNC או SCR.

מנצח מבין הקבוצה הראשונה – פייתון (1.2-1.6 יותר תמציתי מהשפות הFUNC).

מנצח בקבוצה של האחרות – JAVA.

אם חושבים על זה לעומק זה נהדר – במקום 1000 שורות בC ייקח 330 שורות בהאסקל

גודל תוכנית executable

מן הסתם נלקחו רק אלה שהתקמפלו בהצלחה (אחרת לא ניתן לבדוק גודל תוכנית).

מתחלקים ל-2 קבוצות- native code, byte code (הסבר) . שפת RUBY לא נלקחה (אין לה)

<http://programmers.stackexchange.com/questions/24558/is-python-interpreted-or-compiled>

הסבר על BYTE CODE

מושג – static linkage

המפסידה הגדולה : GO

קבוצה של GO וHASKELL הגדולות ביותר – NATIVE גדולים

קבוצה של BYTECODE – קטנים: כל השאר

C – NATIVE קטן איפשהו באמצע (קרוב לBTYECODE)

הבדל גדול יחסית : בין שפות במרכז הסקאלה HASKELL מול JAVA- פי 263.8 בגודל התוכנית !

המנצחת הגדולה: פייתון

זמן ריצה