Programming Languages Midterm Take 2 2022

Name:

**1. Why is it more efficient to use constraints vs procedural programming for the layout of a user interface?**

There are too many layout shapes and human languages to procedurally layout all the possibilities. It is only reasonable to give constrains, min box sizes, what is above what, etc., and let the UI layout engine size and place components according to these constraints.

**2. C++ is called multi-paradigm, because it supports several styles of programming. Explain why the recursive version of mypow(double k, int p) below is more computationally efficient than the procedural version for large p**

|  |  |
| --- | --- |
| **Procedural** | **Functional** |
| double mypow(double k, int p) {  double ans = 1.0;  for (int i=0; i<p; ++i) {  ans = ans \* k;  }  return ans;  } | double mypow (double k, int p) {  if (p == 0) {  return 1.0;  }  double r = mypow(k,p/2);  return (p % 2 == 0) ? r\*r : k\*r\*r;  } |

The procedural version takes p steps through the for-loop. The recursive version keeps dividing p by 2, until p/2^depth is zero (p < 2^depth) so this happens in depth=log2(p)+1 steps. So for p = 1,000,000 the procedural version takes about 1,000,000 steps, while the recursive one takes log2(1,000,000) approx 20 steps, which is about 1,000,000 / 20 = 50,000 times faster.

**3. You are working on a satellite command module. Your team considers receiving messages containing python scripts for position control of the camera system. Are you for or against this idea and why?**

Against. PDF was invented to keep printers from locking up in the other room. So now you want to lock up satellites in orbit? Any general purpose language for this would be a serious risk: how much resources would it use? Could it do something that was not camera related? When can you decide that it is done?

**4. You are designing a language for scripting characters for video games. What language style(s) (procedural, logic, functional, object) make the most sense here?**

Object/logical. The components are aggregate ideas (character, enemy) but the rules are mostly constraints: when the character is close, the enemy is agitated and wants to move toward the character.

**5. Describe the regular expression [0-9A-F]+(.[0-9A-F]+)\* in plain English. Assume ‘.’ is a regular character.**

Non-empty uppercase hexadecimal strings, with as many dots (.) between hexadecimal digits you like (dots can’t be adjacent or start or end the string). AB34AB, AB.34.AB A.B34AB for example.

First, H=[0-9A-F] would match any single uppercase hexadecimal digit.

Next, G=H+ represents 1 or more hexadecimal digits.

Finally G(.G)\* means G followed by 0 or more .G’s: so G, G.G, G.G.G, G.G.G.G, etc. Since each G has at least one hex digit, the dots cannot be next to each other or at either end.

**6. Make a parse tree for “if p then (not a) and (not b) else not (a or b) fi”**

**<if> := if <exp> then <exp> else <exp> fi**

**<var> := a | b | c | …. | z**

**<lit> := true | false**

**<op> := and | or**

**<exp> := <var> | <lit> | ( <exp> ) | <exp> <op> <exp> | not <exp> | <if>**

<if>

if <exp> then <exp> else <exp> fi

not <exp>

<exp> <op> <exp>

<var>

<exp> <op> <exp>

not <exp>

and

not <exp>

p

<var>

<var>

a

b

<var>

a

<var>

b

**I left out the <exp> -> “(“ <exp> “)” transitions and the last <op> -> “or” but that’s the idea.**

**7. Assuming <type> <id> <exp> and <statement> were properly defined, write a grammar which would accept a C-style for loop like “for (int i=0; i<10; ++i) ….”**

**<for> := for ( <statement>; <exp>; <statement> ) <statement-or-block>**

**8. Why is ambiguity a problem with context free grammars?**

If a part of a language can be interpreted in more than one way (like is 3+4\*5+6 = 77 or 29?) then the meaning of the language is unclear both to the typically human author and the typically machine translator.