Linguagens de Programação

Exercícios envolvendo conceitos de linguagens

1) Por é útil que um desenvolvedor tenha alguma experiência no projeto de linguagens, mesmo que ele nunca projete uma linguagem de programação?
Ter experiência no projeto de linguagens é útil por algumas razões, entre elas:

- Compreensão profunda das linguagens, o que pode auxiliar o desenvolvedor a escolher ferramentas e formas de resolver problemas complexos com as linguagens já existentes.
- Resolução de problemas complexos. A experiência nessa área traz o estudo de conceitos de linguagens como expressividade e otimização de código, que ajudam a melhorar a capacidade de resolver problemas complexos.
- Melhorar a qualidade dos códigos desenvolvidos, por causa da melhor compreensão das diferentes características das linguagens de programação.
- Entendimento de compiladores/interpretadores, que pode fazer o programador entender como otimizar seu código, evitando erros comuns relacionados a performances.

Concluindo, a experiência nessa área apenas contribui para as habilidades de programação dos desenvolvedores, além de melhorar a qualidade dos códigos que serão entregues no mercado.

2) Qual a desvantagem de ter recursos demais em uma linguagem?
Quando uma linguagem tem recursos demais, há algumas desvantagens, mesmo que consiga providenciar uma vasta gama de possibilidades. A principal é o tempo que leva para alguém aprende-lá, podendo demorar muito e desestimular programadores a tentarem ela, da mesma forma que a torna intimidadora. Outra desvantagem seriam as revisões e como atualizá-la. Outra desvantagem de uma linguagem com essas características seria a comparação com outras, pois ela teria muitos recursos, porém outras com a mesma capacidade, tendo um foco específico conseguiria superá-la.

3) Como a sobrecarga de operador definida pelo usuário pode prejudicar a legibilidade de um programa?
Prejudica a legibilidade pois não sabemos com tanta facilidade se a implementação do operador é nativa ou não, e temos que gastar mais energia para verificar.

4) Cite um exemplo de falta de ortogonalidade na linguagem C. (dica: structs e vetores...)
A semântica não é sempre a mesma na hora de utilizarmos array com ponteiros, por exemplo:

```
int arr[5] = {1, 2, 3, 4, 5};
int *ptr = arr;

// Os 3 prints abaixo imprimem o 3º elemento do array
printf("%d\n", arr[2]);
printf("%d\n", ptr[2]);
printf("%d\n", 2[arr]);
```

5) Qual linguagem usou ortogonalidade como principal critério de projeto?
Algol 68

6) O que é tratamento de exceções? Qual cuidado que se deve ter ao usar esse recurso?
Tratamento de exceções é uma prática que visa separar 'prever' e reagir de acordo a erros que podem acontecer durante a execução de um programa. Estes erros não impedem a compilação de um código, e podem acontecer ou não dependendo das entradas e variações de um programa. Um bom cuidado para se tomar ao tratar exceções é separá-las bem em diversas categorias visto que se levantarmos um "Erro Genérico" cada vez que uma exceção acontece, pode ser difícil de tratar e definir onde está tal erro.

7) Qual o nome da categoria de linguagens de programação cuja estrutura é ditada pela arquitetura de computadores von Neumann?
Programação Interativa (de Alto Nível)

8) Quais são os três recursos fundamentais da linguagem orientada a objetos?
Abstração, Encapsulamento, Herança e Polimorfismo

9) Qual método produz uma execução de programas mais rápida? Um compilador ou um interpretador puro?
Compilador, visto que analisa o código inteiro de uma vez, enquanto o interpretador deve interpretá-lo cada vez que uma função é chamada, por exemplo.

10) O que faz um ligador?
O ligador une diversos arquivos (de bibliotecas, por exemplo) em um único código objeto executável.

11) O que é o gargalo de von Neumann?
Acontece quando o CPU é mais rápida que a memória, e acaba por vezes por ficar ociosa, por ter de esperar a memória receber ou transferir dados.

12) Classifique as seguintes linguagens quanto a implementação (compilador, interpretador puro, sistema híbrido):
a. C - Compilador
b. C# - Compilador (geralmente compilado para uma linguagem intermediária que é então executada por uma VM)
c. Javascript - Compilada (em tempo de execução, graças a "JIT" introduzida aos navegadores)
d. Java - Sistema Híbrido (compilado para bytecode Java e, em seguida, interpretado por uma VM Java)
e. C++ - Compilador
f. Python - Interpretador (geralmente por meio de CPython, e existem versões que compilam para bytecode, mas geralmente são interpretados)