Nome: Eduardo Henrique de Almeida Izidorio
Matrícula: 2020000315
Disciplina: Linguagens de Programação
1ª Lista de Exercícios de linguagens de programação

1) Por que é útil para um programador ter alguma experiência no projeto de linguagens, mesmo que ele nunca projete uma linguagem de programação?

2) Como o conhecimento de linguagens de programação pode beneficiar toda a comunidade da computação?

3) Que linguagem de programação tem dominado a computação científica nos últimos 50 anos?

4) Que linguagem de programação tem dominado as aplicações de negócios nos últimos 50 anos?
5) Que linguagem de programação tem dominado a Inteligência Artificial nos últimos 50 anos?
6) Em que linguagem o UNIX é escrito?
7) Qual é a desvantagem de ter muitas características em uma linguagem?
8) Cite um exemplo da falta de ortogonalidade no projeto da linguagem C.
9) Qual linguagem usou a ortogonalidade como um critério de projeto primário?
10) Que sentença de controle primitiva é usada para construir sentenças de controle mais complicadas em linguagens que não as têm?
11) Que construção de uma linguagem de programação fornece abstração de processos?
12) O que significa para um programa ser confiável?
13) Por que verificar os tipos dos parâmetros de um subprograma é importante?
14) O que são apelidos?
15) O que é o tratamento de exceções?
16) Por que a legibilidade é importante para a portabilidade de escrita?
17) Como o custo de compiladores para uma linguagem está relacionado ao projeto dela?
18) Qual tem sido a influência mais forte no projeto de linguagens de programação nos últimos 50 anos?
19) Qual é o nome da categoria de linguagens de programação cuja estrutura é ditada pela arquitetura de computadores de Von Neumann?
20) Que duas deficiências das linguagens de programação foram descobertas como um resultado da pesquisa em desenvolvimento de software dos anos 1970?

21) Quais são os três recursos fundamentais de uma linguagem orientada a objetos?
22) Qual foi a primeira linguagem a oferecer suporte aos três recursos fundamentais da programação orientada a objetos?
23) Dê exemplo de dois critérios de projeto de linguagens que estão em conflito direto um com o outro.
24) Quais são os três métodos gerais de implementar uma linguagem de programação?
25) Qual produz uma execução de programas mais rápidas, um compilador ou um interpretador puro?
26) Que papel a tabela de símbolos tem em um compilador?
27) O que faz um ligador?
28) Por que o gargalo de Von Neumann é importante?
29) Quais são as vantagens de implementar uma linguagem com um interpretador puro?

Respostas

1- Por que isso facilita o aumento na capacidade de expressar ideias, assim fazendo com que o programador não tenha tanto dificuldade na elaboração de ideias, ter a facilidade de aprender e entender novas linguagens, aumentar a capacidade de projetar novas linguagens e entedes e saber a importância da implementação da melhor forma possível e ter um avanço geral da computação.

2- Pode beneficiar nas novas linguagens que serem criadas a partir das linguagens mais antigas, utilizando os pontos fortes das mesmas para criar, melhora, mais eficazes e mais facis de ser usadas pela comunidade.

3- Tem sido a FORTRAN, que foi criada para otimizar a velocidade das máquinas, e vem sendo utilizada até hoje mas sofrendo diversas atualizações durante os anos que se passaram.

4- Tem sido a COBOL, que foi criada para o processamento de banco de dados e que é considerado uma excelente linguagem gerado desses Bancos de dados/relatórios.

5- Tem sido a LISP, Criada por John McCarthy, 1958, onde a linguagem utiliza a notação de lambda e trabalha extensivamente com listas e outras estruturas elementares para a aplicação de inteligência.

6- Inicialmente escrita em Assembly, posteriormente reescrita em linguagem C, por Kenneth Thompson, Dennis Ritchie, entre outros, em meados de 1970.

7- Linguagens que utilizam ou que continham muitos recursos, tendem a consumir muito espaço na memória e muito processamento da máquina.

8- Um exemplo em C, para a falta de ortogonalidade, podemos falar de dois tipos de dados estruturados, Vetores e registros, onde os registros podem ser retornados de funções, Vetores já não podem.

9- Assembly utilizando pelas mainframes da IBM, principalmente se comparado com os minicomputadores VAX.

10- Vetores e ponteiros. Estes em grande parte são usados para se construir sentenças mais complexas, exemplo das mesmas, listas encadeadas, arrores e outros.

11- Pode ser encontrado na construção de uma função (Subprograma), a fim de resolver algum problema específico nele. A abstração é algo tido como importante pois permite a facilidade de escrita de uma linguagem, se não for possível ter essa abstração, o código dessa função teria que ser replicado nos diversos pontos do programa onde ele fosse utilizado.

12- Para um programa ser tido como confiável tem que está de acordo com base Verificação de tipos (programa que executa testes para detectar erros de tipo, tanto em tempo de compilação, quanto em tempo de execução) e Tratamento de Exceções (programa que tem a habilidade de identificar e tratar erros, tomando medidas corretivas quando uma falha é encontrada ou uma exceção ocorre).

13- Quando a linguagem ou o ambiente não oferece um tipo de verificação de dados você poderá ter uma inconsistência de dados no final do programa.

14- São identificadores, que geralmente são utilizados quando se é possível ter um ou mais nomes definidos para acessar a mesma célula de memória.

15- O tratamento de exceção nada mais é que a habilidade de um programa de interceptar erros em tempo de execução e resolvê-los.

16- A legibilidade é um dos conceitos mais importantes na hora de avaliar uma linguagem de programação, pois é o quão fácil uma linguagem é de se programar. Quanto maior for sua legibilidade, maior será seu nível de abstração, ela será considerada uma linguagem de alto nível.

17- Dependendo de como é o projeto da linguagem, o compilador pode ou não ter uma grande quantidade de chegagens antes de terminar o processo de compilação. Se o quantidade de chegagens for muito, então ele poderá acabar tornando-se ineficiente.

18- A Arquitetura de computadores de Von Neumann, que possibilita uma máquina armazene seus programas no mesmo espaço de memória que os dados, desse jeito podendo manipular tais programas.

19- São as linguagens imperativas, que são linguagens que descrevem a computação como ações ou comandos que mudam o estado de um programa.

20- As duas deficiências descobertas nesse ano foram a incompletude da verificação de tipos e a inadequação das sentenças de controle (que utilizava muito as estruturas condicionais "goto"; que em muitos casos gerava problema de legibilidade no código.

21- Encapsulamento, herança e polimorfismo.

22- SmallTalK.

23- legibilidade e facilidade de escrita.

24- Compilação, interpretação pura e Métodos Híbridos.

25- Um compilador. Terá uma resposta mais rápida, isso se dá pela rotina que usa a técnica just in time. Já o interpretador puro terá uma execução mais lenta devido no seu tempo de execução ter que fazer vários processos para algo que a máquina entenda.

26- A Tabela de Símbolos serve como uma base de dados para o processo de compilação, onde as informações de Tipos e atributos definido pelo Usuário no programa.

27- O ligador ou também chamado linker é um programa que tem como objetivo coletar programas de sistemas e ligá-los aos programas de Usuário criando assim o módulo de carga.

28- É a principal força limitante da velocidade de computadores Von Neumann, e o gargalo de Von Neumann tem sido um dos principais motivos para a pesquisa e o desenvolvimento de computadores paralelos.

29- Correções e alterações são mais rápidas de serem realizadas; código não precisa ser compilado para ser executado e consomem menos memória.