2º Lista

Aluno: Angelo Almeida Ferro

Curso: Ciências da Computação

Professor: Herbert Rocha

Disciplina: Linguagens De Programação

1º)

A programação funcional baseia-se no conceito matemático de função, em que para cada elemento do seu conjunto domínio (entrada) há apenas um elemento no seu conjunto contradomínio (saída). Além disso, as funções são normalmente expressas por meio de outras funções - de modo que obter o valor da função para um determinado conjunto de parâmetros envolve não só aplicar as regras daquela função, mas também fazer uso de outras funções.

Por essa razão, a programação funcional não possui o conceito de "variáveis de estado"; se um valor qualquer (seja ele numérico, textual, um vetor, matriz, ou algo mais complexo) aparece num momento qualquer da computação, considera-se que aquele valor é único e imutável durante todo o processo. Pode-se estabelecer relações entre valores - como dizer que 2 é a metade de 4 - mas o valor 2 sempre será um 2 e o 4 sempre será um 4.

VANTAGENS:

- Um alto nível de abstração, especialmente quando as funções são utilizadas, suprimindo muitos detalhes da programação e minimizando a probabilidade da ocorrência de muitas classes de erros.
- A não dependência das operações de atribuição permite aos programas avaliações nas mais diferentes ordens. Esta característica de avaliação independente da ordem torna as linguagens funcionais as mais indicadas para a programação de computadores maciçamente paralelos.
- A ausência de operações de atribuição torna os programas funcionais muito mais simples para provas e análises matemáticas do que os programas procedurais.

DESVANAGENS:

- Menor eficiência.
- Problemas que envolvam muitas variáveis (ex. contas de banco) ou muitas atividades sequenciais são muitas vezes mais fáceis de se trabalhar com programas procedurais ou programas orientados a objeto.

2º)

3º)

4º)

Na programação imperativa considera-se que as estruturas de dados (de novo, números, texto, vetores, matrizes, etc) existem em um determinado estado, e que através de comandos específicos para o computador esse estado vai sendo alterado no decorrer do programa, de modo que ao final da execução ele represente as consequências de um processo lógico. Os comandos disponíveis são fixos pela linguagem, e normalmente há comandos distintos para tratar de tipos de dados distintos.

O conceito de "objeto" é uma extensão desse comportamento para além dos tipos de dados da própria linguagem, aplicando-os também aos tipos definidos pelo usuário: para cada estrutura de dados definida, define-se também uma série de operações que podem ser aplicadas a essa estrutura - em geral influenciadas pelo seu estado interno - e a partir de então permite-se usar essas estruturas como tipos distintos, e não como estruturas de dados genéricas. Some-se a isso o conceito de "tipos mais genéricos" e "tipos mais específicos" (i.e, herança), e temos a possibilidade de criar uma taxonomia rica de tipos cada um com suas operações - de modo que os programas possam ser expressos num nível de abstração mais alto do que aqueles que utilizem somente os tipos fornecidos pela linguagem de programação em si.

Quando os objetos e suas relações são o ponto central de uma linguagem de programação, diz-se que ela é "orientada por objetos" pois todo o fluxo lógico do programa se baseia em quais objetos existem e como eles interagem entre si. Ou seja, inicialmente esses objetos estão lá - em seu estado inicial - e à medida que o programa interage com o seu ambiente (seja através de uma entrada, ou através da interação com o usuário) esses objetos vão mudando de estado.

BENEFICIOS:

- Torna mais rápidas as atividades de programação e manutenção de sistemas de informação;
- Tem caráter unificador: trata todas as etapas do desenvolvimento de sistemas e ambientes sob uma única abordagem;
- Reusabilidade de código
- Escalabilidade de aplicações
- Mantenabilidade
- Apropriação

5º)

```
public class Car
{
  int year;
  string make;
  double speed;

  public Car(int y, String m, double beginningSpeed)
  {
    year = y;
  }

  public int getYear()
  {
    int tmp = year;
    Roda r = new Roda(tmp);
    return year;
  }
}
```

TIPOS:

Atributos:

```
int year;
string make;
double speed;
```

Parametros:

```
(int y, String m, double beginningSpeed)
```

Variável Local:

```
int tmp = year;
Roda r = new Roda(tmp);
return year;
```

6º)

7º)

A)

B)

JAVA:

```
public class par{
   public static void main(String[] args){
      Scanner input = new Scanner(System.in);
      System.out.print("Informe um valor inteiro: ");
      int num = input.nextInt();

      if(num % 2 == 0)
        System.out.print("Você informou um numero par");
      else
        System.out.print("Você informou um numero impar");

      System.exit(0);
    }
}
```

COMPILAÇÃO:

Abra o terminal escreva mkdir lista2 e aperte enter.

Escreva cd lista2 e aperte enter.

Escreva touch questao7b.java e aperte enter.

Procure o a pasta que você criou com o nome de lista2 e então abra ela e clicke com o botão direito no arquivo questao7b.java e depois clicke em abrir com o geany.

Então copie o código java acima e cole no geany e então salve segurando crtl e apertando S ou clickando em arquivo e em salvar.

Volte para o terminal e escreva javac questao7b.java e aperte enter.

E então escreva java questao7b e aperte enter.

HASKELL:

```
soma1 = (+) 1
numero_par n
  | n `mod` 2 == 0 = True
  | otherwise    = False
```

COMPILAÇÃO:

Escreva touch questao7b.hs e aperte enter.

Clicke com o botão direito no arquivo questao7b.hs e depois clicke em abrir com o geany.

Então copie o código haskell acima e cole no geany e então salve segurando crtl e apertando S ou clickando em arquivo e em salvar.

Volte para o terminal e escreva ghci questao7b.hs e aperte enter.

C)

D)