Estruturas de dados heterogêneas

Registros

Vimos inúmeras aplicações onde são necessários conjuntos de elementos do mesmo tipo, e para tal utilizamos os vetores.

No entanto, em alguns problemas há necessidade de definirmos conjuntos onde os elementos não sejam do mesmo tipo.

Um típico exemplo de nosso cotidiano é a utilização do conjunto de informações que caracterizam um aluno: Nome(caractere), CPF(inteiro), RG(inteiro), data de nascimento(caractere), coeficiente de rendimento(real), etc..

UNIVASF

Estruturas de dados heterogêneas

Em uma análise superficial um estudante poderia pensar que uma solução para a questão apresentada poderia ser obtida declarando-se cinco variáveis:

algoritmo "exemplo"

var Nome: caractere

CPF: inteiro

RG: inteiro

data_de_nascimento: caractere

coeficiente_de_rendimento: real

...

Para uma melhor visualização da utilidade dos registros basta imaginarmos que ao invés de manipular as informações de um aluno exista a necessidade de gerenciamento de uma turma com cinquenta alunos.

Estruturas de dados heterogêneas

Um estudante desatento imaginaria ser necessário a declaração de 250 variáveis. Porém, um estudante com uma visão mais coerente dos conceitos estudados sugeriria a utilização de cinco vetores:

algoritmo "exemplo"
var Nomes: vetor [1..50] de caractere
CPFs: vetor [1..50] de inteiro
RGs: vetor [1..50] de inteiro
datas_de_nascimento: vetor [1..50] de caractere
coeficientes_de_rendimento: vetor [1..50] de real
...

Porém, manipular de forma adequada os vetores, mantendo seus dados consistentes, se torna trabalhoso. Com a utilização de um registro podemos resolver este problema apenas com um vetor de cinquenta registros.

Estruturas de dados heterogêneas

Exemplo:

```
algoritmo “exemplo”
var alunos: vetor [1..50] de registro
      inicio
         nome: caractere
         CPF:  inteiro
         RG:  inteiro
         datas_de_nascimento: caractere
         coeficientes_de_rendimento: real
      fimregistro
...
```

A cada um dos elementos que constituem um registro é dado o nome de campo. No exemplo acima, temos os campos: nome, CPF, RG, datas_de_nascimento e coeficientes_de_rendimento.

Estruturas de dados heterogêneas

Com base no exemplo anterior podemos deduzir a estrutura geral para a declaração de um registro:

<nome_da_variavel>: registro
inicio
<nome_campo_1>: <tipo_campo_1>
<nome_campo_2>: <tipo_campo_2>
...
<nome_campo_n>: <tipo_campo_n>
fimregistro

Estruturas de dados heterogêneas

Abriremos um parêntese em nosso estudo sobre registros para falarmos sobre **definição de tipo de dado**.

Com o objetivo de facilitar a leitura e consequentemente o entendimento dos algoritmos construídos foi criada o conceito de definição de tipo de dado.

Sintaxe:

tipo <nome_do_tipo>: <definicao_do_tipo>

Exemplo:

tipo vetor_de_inteiros: vetor [1..100] de inteiro

As definições de tipos devem ser feitas entre a constante caractere que nomeia o algoritmo e a declaração de variáveis globais ou dos módulos.

Obs.: Um estudante atento já vislumbrou as vantagens da definição de tipo na passagem de vetores e registros como parâmetros em módulos.

Estruturas de dados heterogêneas

Com a utilização dos conceitos vistos podemos resolver o problema aludido da seguinte forma:

```
algoritmo “exemplo”
tipo registro_aluno: registro
inicio
   nome: caractere
   CPF:  inteiro
   RG:  inteiro
   datas_de_nascimento: caractere
   coeficientes_de_rendimento: real
fimregistro
tipo vetor_de_registros: vetor [1..50] de registro_aluno
var alunos: vetor_de_registros
...
```

Estruturas de dados heterogêneas

Agora devemos tratar de como é feita a manipulação de um registro.

Da mesma forma que trabalhamos com um vetor acessando-o elemento a elemento, seja para atribuição ou seja para consulta de um valor, o mesmo ocorre com relação aos registros, devemos acessá-lo campo a campo.

Para acessarmos um determinado campo de um registro devemos utilizar o operador "." da seguinte forma:

No caso do exemplo com o qual temos trabalhado, a leitura do campo nome do décimo segundo aluno da turma é feita através de

leia(alunos[12].nome)

ou a impressão na saída padrão do CPF do terceiro aluno seria feita da seguinte forma

escreva(alunos[3].cpf)

Exercício 43:

Defina um tipo de dado capaz de armazenar as seguintes informações sobre um determinado cliente de um banco: nome, CPF, RG, número da conta, data de abertura da conta e saldo.

```
tipo data: registro
inicio
   dia:inteiro
   mes: inteiro
   ano: inteiro
fimregistro
tipo registro_conta: registro
inicio
   nome: caractere
   cpf: caractere
   rg: caractere
   numero_conta: inteiro
   data_abertura: data
   saldo: real
fimregistro
```

Exercício 44:

Com base no exercício anterior, construa um algoritmo que manipule um vetor com 15 registros de clientes, onde cada registro é um elemento do tipo de dado definido. A manipulação do vetor é feita através dos seguintes módulos: inicializar vetor, imprimir um determinado registro com base no valor do campo CPF e imprimir um determinado registro com base em sua posição no vetor. O algoritmo deve se utilizar de forma satisfatória dos módulos mencionados e não deve possuir variáveis globais.

UNIVASF

```
algoritmo "Exercício sobre registros"
  tipo data: registro
  inicio
    dia:inteiro
    mes: inteiro
    ano: inteiro
  fimregistro
  tipo registro_conta: registro
  inicio
    nome: caractere
    cpf: caractere
    rg: caractere
    numero_conta: inteiro
    data_abertura: data
    saldo: real
  fimregistro
  tipo vetor_de_registros: vetor [1..15] de registro_conta
```

```
procedimento inicializar_vetor (var v: vetor_de_registros)
var i: inteiro
inicio
   para i de 1 ate 15 faca
      escreva ("Entre com as informações do registro número ")
      escreval (i, ".")
      escreva ("Número da conta: ")
      leia (v[i].numero_conta)
      escreva ("Nome do cliente: ")
      leia (v[i].nome)
      escreva("CPF: ")
      leia (v[i].cpf)
      escreva("RG: ")
      leia (v[i].rg)
      escreval("Data de abertura: ")
      escreva ("Ano: ")
      leia (v[i].data_abertura.ano)
      escreva ("Mês: ")
      leia (v[i].data_abertura.mes)
      escreva ("Dia: ")
      leia (v[i].data_abertura.dia)
      escreva("Saldo: ")
      leia (v[i].saldo)
   fimpara
fimprocedimento
```

```
funcao imprimir_registro_CPF (var v: vetor_de_registros;
   cpf: caractere):logico
var i: inteiro
inicio
   para i de 1 ate 15 faca
      se (v[i].cpf=cpf) entao
         escreval ("Registro da conta número ", v[i].numero_conta)
         escreval ("Cliente: ",v[i].nome)
         escreval("CPF: ",v[i].cpf)
         escreval("RG: ",v[i].rg)
         escreva ("Data de abertura: ",v[i].data_abertura.dia,"/")
         escreval (v[i].data_abertura.mes, "/", v[i].data_abertura.ano)
         escreval("Saldo: ",v[i].saldo)
         retorne (verdadeiro)
      fimse
   fimpara
   retorne (falso)
fimfuncao
```

```
funcao imprimir_registro_posicao (var v: vetor_de_registros;
   posicao: inteiro):logico
inicio
   se (posicao<1 ou posicao>15) entao
      retorne (falso)
   senao
      escreva ("Registro da conta número ")
      escreval (v[posicao].numero_conta)
      escreval ("Cliente: ",v[posicao].nome)
      escreval("CPF: ",v[posicao].cpf)
      escreval("RG: ",v[posicao].rg)
      escreva("Data de abertura: ",v[posicao].data_abertura.dia,"/")
      escreval ([posicao].data_abertura.mes,"/")
      escreval (v[posicao].data_abertura.ano)
      escreval("Saldo: ",v[posicao].saldo)
      retorne (verdadeiro)
   fimse
fimfuncao
```

```
  procedimento principal()
  var vet_reg: vetor_de_registros
      cpf: caractere
      p: inteiro
  inicio
    inicializar_vetor(vet_reg)
    escreva ("Entre com um CPF para impressão do registro da ")
    escreval ("conta correspondente: ")
    leia (cpf)
    se (nao imprimir_registro_CPF (vet_reg, cpf)) entao
      escreva ("Não existe nenhum conta de cliente com o CPF ")
      escreval ("especificado.")
    fimse
    escreval ("Entre com a posição do registro da conta para ")
    escreval ("impressão, intervalo [1,15]: ")
    leia (p)
    se (nao imprimir_registro_posicao (vet_reg, p)) entao
      escreval ("Posição inválida.")
    fimse
  fimprocedimento
inicio
  principal()
fimalgoritmo
```

Exercício 45:

A solução apresentada para o exercício anterior, visando não tirar o foco da manipulação de registros, não tratou do aspecto extremamente relevante de validação de entradas. Sendo assim faça as alterações necessárias na solução exposta para que esta trate de forma coerente da questão de validação das entradas.