Vetor

Prof. Edwar Saliba Júnior

Janeiro de 2009

Problema

- Preciso de um *software* que armazene a idade, em anos (número inteiro), de quatro pessoas:
  - Maria
  - José
  - Pedro
  - Luiza (que já voltou do Canadá);
- Neste caso, precisamos então criar quatro lugares (posições de memória) para guardar, a idade de cada pessoa.

Para resolver o problema podemos:

- Criar quatro variáveis do tipo inteiro e atribuir a idade de cada pessoa em cada uma das variáveis criadas;
- Exemplo:
  - Idade_Maria ← 13
  - Idade_Pedro ← 50
  - Idade_Jose ← 29
  - Idade_Luiza ← 19

Algoritmo

```
início
   declare Idade_Maria, Idade_Jose, Idade_Pedro, Idade_Luiza :
   inteiro
   Idade_Maria ← 21
   Idade_Jose ←  42
   Idade_Pedro ← 55
   Idade_Luiza ← 10
   escreva "Idade de Maria: ", Idade_Maria
   escreva "Idade de José: ", Idade_Jose
   escreva "Idade de Pedro: ", Idade_Pedro
   escreva "Idade de Luiza: ", Idade_Luiza
fim
```

Mas e se tivermos que guardar 1000 idades. O que fazer?

- Criar 1000 variáveis?

Outro problema:

- Preciso guardar 1000 nomes, 1000 idades, 1000 salários, 1000 endereços e 1000 telefones.

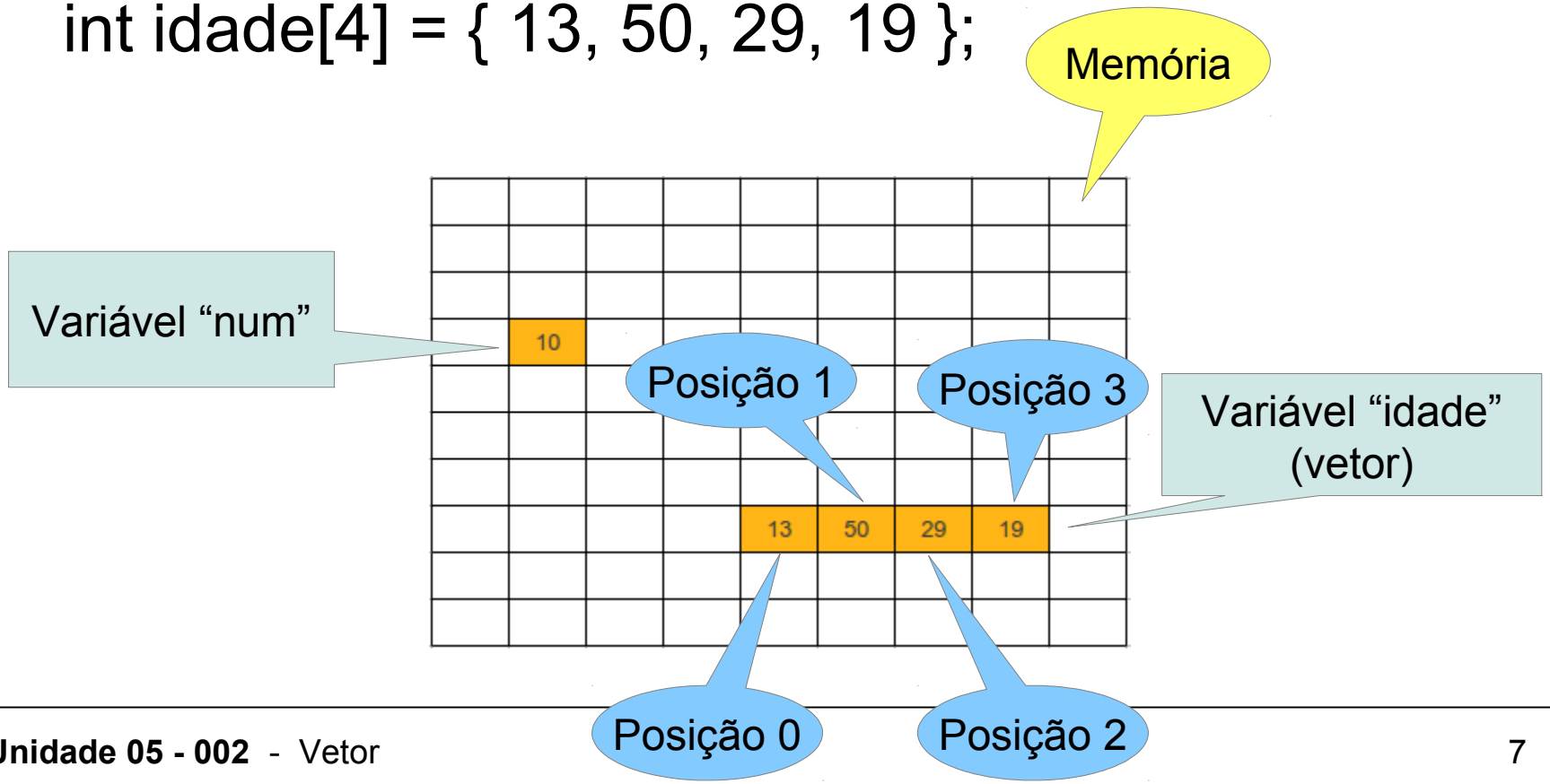
- O que fazer?

Vetor

- Para situações como esta, apresentada anteriormente, foi criada uma estrutura conhecida como VETOR;
- Um vetor nada mais é do que UMA variável com diversas posições de memória numeradas. Onde pode-se guardar diversos valores do mesmo tipo (um em cada posição).

Variável e Memória

```
int num = 10;
int idade[4] = { 13, 50, 29, 19 };
```



Associação

- Deve-se fazer, internamente, uma associação das posições de memória à cada pessoa;

Pessoa	Posição de Armazenagem
Maria	0
José	1
Pedro	2
Luiza	3

Algoritmo

Quantidade de idades

```
início
    declare Idade[4] : inteiro
    Idade[0] ← 13
    Idade[1] ← 29
    Idade[2] ← 50
    Idade[3] ← 19
    escreva "Idade de Maria: ", Idade[0]
    escreva "Idade de José: ", Idade[1]
    escreva "Idade de Pedro: ", Idade[2]
    escreva "Idade de Luiza: ", Idade[3]
fim
```

Posição de Armazenagem

Algoritmo – Entrada Via Teclado

```
início
   declare Idade[4] : inteiro
   escreva "Digite a idade de Maria: "
   leia Idade[0]
   escreva "Digite a idade de José: "
   leia Idade[1]
   escreva "Digite a idade de Pedro: "
   leia Idade[2]
   escreva "Digite a idade de Luiza: "
   leia Idade[3]

   escreva "Idade de Maria: ", Idade[0]
   escreva "Idade de José: ", Idade[1]
   escreva "Idade de Pedro: ", Idade[2]
   escreva "Idade de Luiza: ", Idade[3]
fim
```

Algoritmo – Entrada Via Teclado (Loop)

```
início
   declare Idade[4], cont : inteiro

   para cont ← 0 até 3 passo 1 faça
      escreva "Digite a idade: "
      leia Idade[cont]
   fim para

   cont ← 0
   enquanto (cont < 4) faça
      escreva "Idade: ", Idade[cont]
      cont ← cont + 1
   fim enquanto
fim
```

Algoritmo – Entrada Via Teclado (Loop)

- Vamos resolver o problema proposto anteriormente. Ou seja, um algoritmo para guardar e imprimir 1000 idades distintas:

```
início
   declare Idade[1000], cont : inteiro
   para cont ◄— 0 até 999 passo 1 faça
      escreva "Digite a idade: "
      leia Idade[cont]
   fim para
   cont ◄— 0
   enquanto (cont < 1000) faça
      escreva "Idade: ", Idade[cont]
      cont ◄— cont + 1
   fim enquanto
fim
```

Outro Exemplo

- Ler as notas dos 20 alunos de uma turma (notas de 0 a 100, sem casas decimais), armazená-las em um vetor para posteriormente calcular e imprimir a média da turma.

Resolução em Linguagem C

```
#include <stdio.h>

int main()
{
    int nota[20], total = 0, cont;

    for (cont = 0; cont < 20; cont++) {
       printf("\nDigite a nota: ");
       scanf("%d", &nota[cont]);
    }

    for (cont = 0; cont < 20; cont++) {
       total = total + nota[cont];
    }

    printf("\nA média da turma é: %f", total / 20);

    return 0;
}
```

Atenção! Perigo!

```
#include <stdio.h>

int main()
{
    int vetor[10];

    vetor[100] = 20;

    printf("%d", vetor[100]);

    return 0;
}
```

- Algumas linguagens de programação, como é o caso da "Linguagem C", deixa por conta do programador a tarefa de respeitar os limites que ele mesmo criou para as variáveis.

Strings em Linguagem C

- Implementação de variáveis do tipo String (cadeia de caracteres) em C:

```
#include <stdio.h>

int main()
{
  char Nome[10];
  int i;

  printf("Digite o nome: ”);
  gets(Nome);

  printf("\n\n\nNome digitado: ”);
  for (i = 0; i < 10; i++)
     printf(“\n %c”, Nome[i]);

  return 0;
}
```

Problema: Se o nome digitado tiver menos de 10 caracteres!

Strings em Linguagem C

- Implementação de variáveis do tipo String (cadeia de caracteres) em C:

```
#include <stdio.h>

int main()
{
  char Nome[10];
  int i;

  printf("Digite o nome: ”);
  gets(Nome);
  printf("\n\n\nNome digitado:”);
  i = 0;
  while ((i < 10) && (Nome[i] != '\0'))
  {
    printf(“%c”, Nome[i]);
    i++;
  }

  return 0;
}
```

SOLUÇÃO!

Funções para Tratamento de Strings em Linguagem C

- **Algumas funções para manipulação de strings em C:**
  - **strlen ():** Número de caracteres antes do '\0'
    - Ex: int len = strlen(Nome);
  - **strcpy ():** atribui a uma variável do tipo string uma constante ou o valor de outra string;
    - Ex: strcpy(Nome2, Nome1);
  - **strcmp ()**:
    - Ex: int result = strcmp(Nome1, Nome2)
      - Pode retornar: maior que 0 (Nome1 maior que Nome2), 0 (Nome1 igual a Nome2) ou menor que 0 (Nome1 menor que Nome2);
  - **strcat ():** concatenação;
    - Ex: strcat("saudacoes ", Nome);

Bibliografia

- ASCENCIO, Ana F. G.; CAMPOS, Edilene A. V. ***Fundamentos da Programação de Computadores***. 2ª. ed., São Paulo: Pearson-Prentice Hall, 2007.
- SILVA, Guilherme Baião S. *Slides* da disciplina de Fundamentos de Programação de Computadores I e II. Faculdade INED, 2006.