Lógica de programação

Vetores

Walisson Pereira

walisson_pereira@uvanet.br
Universidade Estadual Vale do Acaraú

Roteiro

Preâmbulo

Preâmbulo

Exercício:

Leia as notas de uma turma de cinco estudantes e depois imprima as notas que são maiores do que a média da turma.

Preâmbulo

Uma possível solução seria:

```c
#include <stdio.h>
int main () {
    float n1, n2, n3, n4, n5, media;
    printf("Digite a nota de 5 estudantes: ");
    scanf("%f %f %f %f %f", &n1, &n2, &n3, &n4, &n5);
    media = (n1 + n2 + n3 + n4 + n5) / 5;
    if (n1 > media)
      printf ("nota: %.1f\n", n1);
    if (n2 > media)
      printf ("nota: %.1f\n", n2);
    if (n3 > media)
      printf ("nota: %.1f\n", n3);
    if (n4 > media)
      printf ("nota: %.1f\n", n4);
    if (n5 > media)
      printf ("nota: %.1f\n", n5);
}
```

Preâmbulo

O programa anterior representa uma solução possível para o problema. O inconveniente dessa solução é a grande quantidade de variáveis para gerenciar e o uso repetitivo de comandos praticamente idênticos, como é o caso dos **ifs**.

Observe:
Esta solução é inviável para uma turma de 100 alunos.

Preâmbulo

Expandir o programa anterior para trabalhar com uma turma de 100 alunos significaria, basicamente, aumentar o número de variáveis para guardar as notas de cada aluno e repetir, ainda mais, um conjunto de comandos praticamente idênticos (os **ifs**). Desse modo, teríamos:

- Uma variável para armazenar a nota de cada aluno: 100 variáveis.
- Um comando de leitura para cada nota: 100 **scanf()**.
- Um somatório de 100 notas.
- Um comando de teste para cada aluno: 100 comandos **if**.
- Um comando de impressão na tela para cada aluno: 100 **printf()**.

Como se pode notar, temos uma solução extremamente engessada para o nosso problema. Modificar o número de alunos usado pelo programa implica reescrever todo o código repetindo comandos praticamente idênticos. Além, da grande quantidade de variáveis para gerenciar tornar mais difícil de ser realizada sem a ocorrência de erros.

Preâmbulo

Como as variáveis do programa têm relação entre si (todas armazenam notas de alunos), podemos declará-las como um único nome para todos os 100 alunos.

Vetor

Vetor

Vetor também é conhecido como variável composta homogênea unidimensional. Isso quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas sequencialmente na memória. Como as variáveis têm o mesmo nome, o que as distingue é um índice que referencia sua localização dentro da estrutura.

Na computação, é comum usarmos o termo **array** (arranjo) para denominar um vetor.

Vetor

Na linguagem C, o vetor é definido da seguinte maneira:

```
<tipoDosDados> <nomeDoVetor> [tamanhoDoVetor];
```

Onde:

- **tipoDosDados** representa o tipo dos dados dos elementos que são armazenados no vetor;
- **nomeDoVetor** é o nome pelo qual o vetor será referenciado; e
- **tamanhoDoVetor** é o número de elementos que podem ser armazenados no vetor.

Vetor

Declaração de um vetor:

```
1 int vet[5];
```

O vetor chamado **vet** possui cinco posições, começando pela posição 0 e indo até a posição 4.

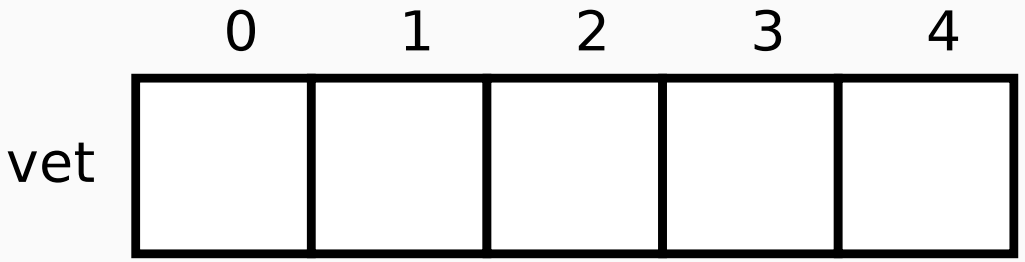

Figura 1: O vetor vet possui a capacidade de armazenar 5 valores inteiros

Vetor

Atribuir valores:

```
1 vet[0] = 2;
2 vet[1] = 4;
3 vet[2] = 6;
4 vet[3] = 8;
5 vet[4] = 10;
```

O vetor chamado **vet** possui cinco posições, começando pela posição 0 e indo até a posição 4.

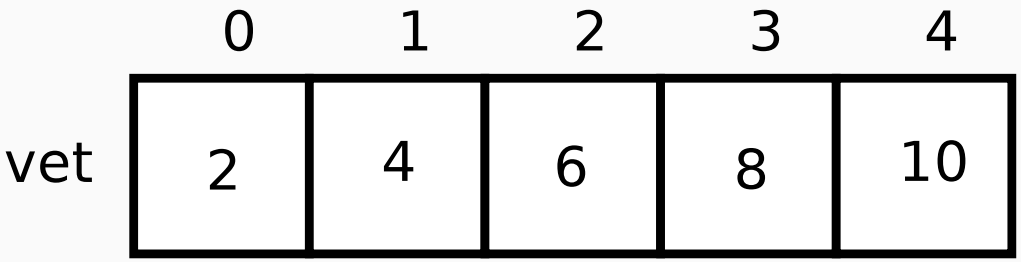

Figura 2: vetor vet preenchido

Vetor

Como acessar valores de um vetor:

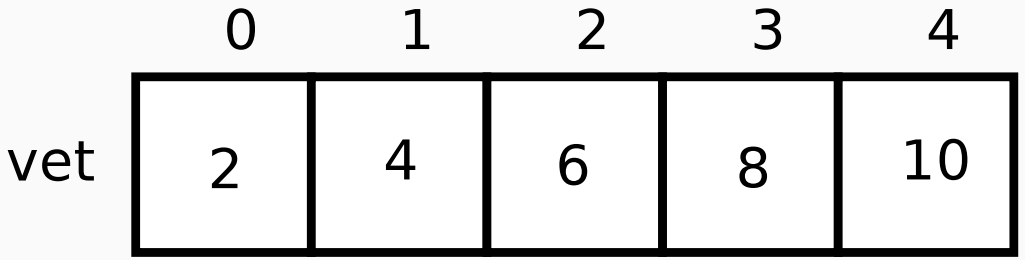

Figura 3: vetor vet preenchido

```
1 printf("%d\n", vet[1]);
2 printf("%d\n", vet[3]);
```

Terminal:

```
1 4
2 8
```

Voltando ao exercício inicial:

Leia as notas de uma turma de cinco estudantes e depois imprima as notas que são maiores do que a média da turma.

Vetor

Uma nova possível solução seria:

```c
#include <stdio.h>
int main () {
  float nota[5];
  printf("Digite a nota de 5 estudantes: ");
  for (int i = 0; i < 5; i++)
    scanf("%f", &nota[i]);
  float media = 0;
  for (int i = 0; i < 5; i++)
      media += nota[i];
  media /= 5;
  for (int i = 0; i < 5; i++)
    if (nota[i] > media)
      printf("nota: %.1f\n", nota[i]);
}
```

Vetor

Como atribuir valores a um vetor a partir do teclado:

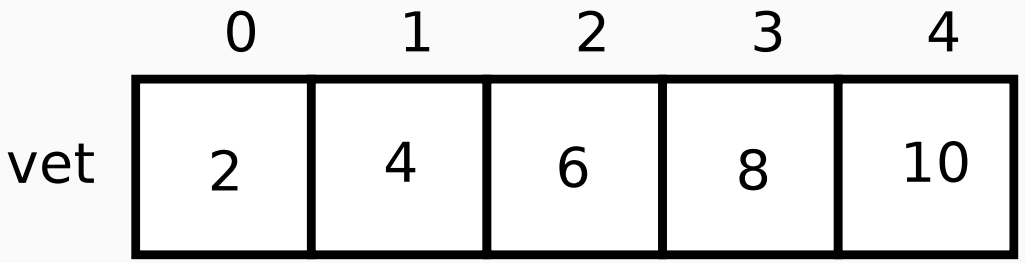

Figura 4: vetor vet preenchido

```
1 scanf("%d", &vet[3]);
2 printf("%d\n", vet[3]);
```

Terminal:

```
1 15
2 15
```

Obs: você deve inserir os dados posição por posição.

Vetor

Exemplo: Calcule a média aritmética de 5 notas já informadas.

```c
#include <stdio.h>
int main () {
    float notas[] = {7, 5, 6, 9, 8};
    float soma, media;
    int x = 0;
    while (x < 5) {
        soma += notas[x];
        x += 1;
    }
    media = soma / 5;
    printf("Media: %.2f\n", media);
}
```

Vetor

Exemplo: Leia e depois imprima 5 notas.

```c
#include <stdio.h>
int main () {
    float notas[] = {0, 0, 0, 0, 0};
    float soma, media;
    int i = 0;
    while (i < 5) {
        scanf("%f", &notas[i]);
        i++;
    }
    printf("As notas digitadas foram: ");
    i = 0;
    while (i < 4) {
        printf("%.2f, ", notas[i]);
        i++;
    }
    printf("%.2f\n", notas[i]);
}
```

Vetor

Resumo das formas de declarar um vetor:

```
1 int numeros[10];
```

Cria um vetor com 10 posições de inteiro, mas nada pode se afirmar sobre os valores contido nele.

```
1 int numeros[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Cria um vetor com 10 posições de inteiros contendo exatamente os valores contido entre chaves. Não é necessário informar o tamanho do vetor. Ele terá o tamanho do conjunto informado.

Strings

O vetor de elementos do tipo **char** pode ser considerado o tipo **string**. Strings correspondem a uma sequência de caracteres. Geralmente são usadas para realizar a entrada e saída de dados em um programa e para armazenar dados não numéricos.

Strings

É comum representar uma string como um vetor de caracteres (tipo **char**), dispondo das mesmas operações comuns a essa estrutura de dados.

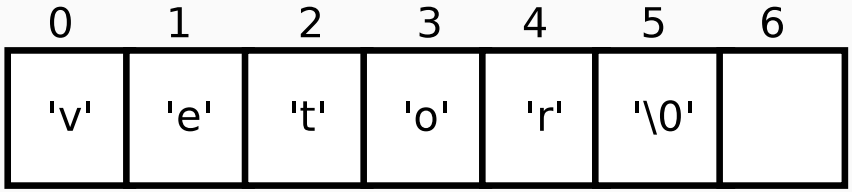

Figura 5: Caption

Toda string armazenada em um vetor é terminada por um caractere especial, conhecido como caracter nulo, ‘\0‘.

A principal função desse caractere é indicar o fim de uma string, evitando o acesso indesejado a uma posição do vetor que se encontra 'vazia'.

Strings

Na linguagem C, não existe operações básicas para a manipulação de vetores como um todo. Você precisa fazer a manipulação item por item do vetor.

Por este motivo, após criada uma string (ou um vetor de char), não é possível adicionar valores com uma simples atribuição. A exceção é quando você já cria o vetor com um valor. Neste caso, o compilador se encarrega de já colocar os valores nas posições especificadas.

```
1 char nome[30];
2 nome = "juca"; //isto nao funciona
3 char nome_do_meio[30] = "mendes"; //isto funciona
4 char sobrenome[] = "da silva"; //isto funciona
```

Strings

Podemos usar algumas funções da biblioteca **string** para manipular um vetor do tipo **char**.

- **strcpy(destino, fonte)** realiza uma cópia da string.
- **strlen(palavra)** calcula o tamanho da string informada.
- **strcmp(palavra1, palavra2)** compara se as duas strings são iguais. Caso sejam iguais, retorna o valor zero. Caso contrário, retorna um valor diferente de zero.

Faça o teste:

```c
#include <stdio.h>
#include <string.h>

int main () {
    char nome[30];
    strcpy(nome, "maria");
    int tamanho = strlen(nome);
    printf("A string %s possui tamanho %d\n", nome, tamanho);
    int comparacao = strcmp(nome, "mara");
    printf("O resultado da comparacao = %d\n", comparacao);
    comparacao = strcmp(nome, "maria");
    printf("O resultado da comparacao = %d\n", comparacao);
    return 0;
}
```

Pesquisa

Podemos pesquisar se um elemento está ou não em uma lista, verificando do primeiro ao último elemento se o valor procurado estiver presente:

Pesquisa

```c
#include <stdio.h>
int main () {
  int vet[] = {15, 7, 27, 39};
  int pesquisar, achou = 0, i = 0, tamanho = 4;
  scanf("%d", &pesquisar);
  while (i < tamanho) {
    if (vet[i] == pesquisar) {
      achou = 1;
      break;
    }
    i++;
  }
  if (achou)
    printf("%d achado na posicao %d\n", pesquisar, i);
  else
    printf("%d nao encontrado\n", pesquisar);
}
```

Exercícios

Pesquisa

Exercício:

1. Faça um programa que:
   - Leia N números reais (considerar N menor ou igual a 100).
   - Calcule a média dos elementos de índices ímpares e a média dos elementos de índices pares.
   - Apresente os elementos que têm valor superior à sua média.
2. Faça um programa que:
   - Leia uma frase com até N caracteres (considerar N menor que 100).
   - Apresente o total de vogais que existem na frase.

Referências

Referências

1 ASCENCIO, A.F.G.; CAMPOS, E.A.V. de. Fundamentos da Programção de Computadores. Algoritmos, Pascal e C/C++. São Paulo: Pearson Prentice Hall, 2012.

2 VAREJÃO, F. M. V. Introdução à programação: uma nova abordagem usando C. Rio de Janeiro: Elsevier, 2015.

3 BACKES, A. Linguagem C: completa e descomplicada. Rio de Janeiro: Elsevier, 2013.