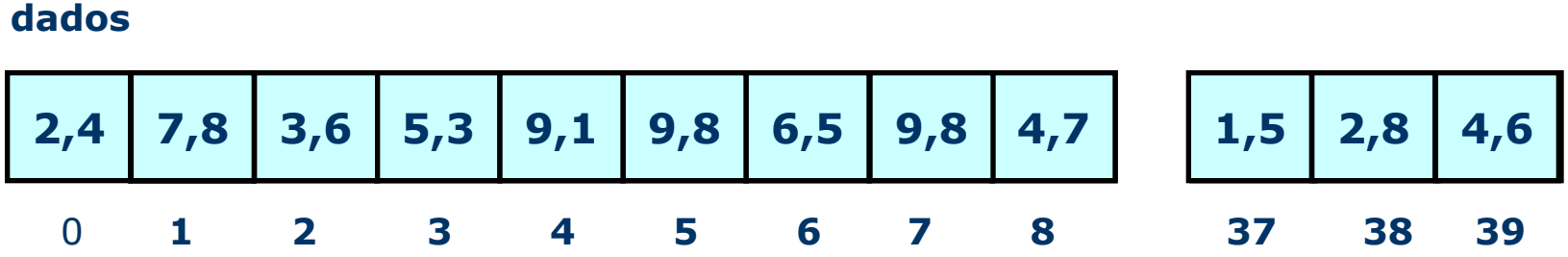
PCI- Vetores

Profa. Mercedes Gonzales Márquez

Conceito

- Sequência de valores todos do mesmo tipo
- Nome único para a variável
- Acesso por índice
- Tamanho fixo
- Numeração de 0 até tamanho-1
- Alocados sequencialmente na memória
- Exemplo: Um vetor com nome “dados” de 40 posições reais.



Declaração

Declaração de vetor:
Tipo variavel [tamanho];
Onde
Tipo é qualquer tipo de dado da linguagem C
Variável é o Nome da variável e
Tamanho é o número de elementos.

Exemplos:
int vetor[40];
double dados[100];

Acesso

- Para manipular (atribuir, ler, escrever) os elementos de um vetor devemos especificar a sua posição.

- A posição do vetor é determinada por meio de uma constante, de uma expressão aritmética ou de uma variável que estiver dentro dos colchetes. Ela é também chamada de índice.

Exemplo:

```
int vetor[10],i=1;
vetor[5] = 3; vetor[1] = 5; vetor[2] = 7; vetor[3]=1;
vetor[0] = vetor[1] + vetor[2];
printf (“O elemento eh %d”, vetor[vetor[i+4]]);
```

Exemplo

Exemplo 1: Ler 10 notas, calcular a média das 10 notas e imprimir as notas maiores ou iguais a média.

```
int main(){
float notas[10], media=0.0;
int i;
for (i = 0; i<10; i++){
        printf(“Digite nota %d: ”, i+1);
        scanf(“%f”, &notas[i]);
        media = media + notas[i];
}
media = media / 10.0; printf(“Media = %f\n”, media);
```

Exemplos

Exemplo 1: Ler 10 notas, calcular a média das 10 notas e imprimir as notas maiores ou iguais a média.

Continuacao:

```
        for (i = 0; i<10; i++)
            if (notas[i]>= media)
                printf("Nota %d: %f\n", i + 1, notas[i]);
}
```

Exemplos

Exemplo 2. Programa das notas utilizando vetores, número de notas variável:

```
#include <stdio.h>
#define MAX 20
int main(){
float notas[MAX], media=0.0;
int i, n;
do {
  printf(“Quantas notas ser˜ao digitadas? ”);
  scanf(“%d”, &n);
}while ((n<= 0) || (n > MAX));
```

Exemplos

Exemplo 2. Programa das notas utilizando vetores, número de notas variável:

Continuaçao:

```
for (i = 0; i<n; i++){
    printf(“Digite nota %d: ”, i+1); scanf(“%f”, &notas[i]);
    media = media + notas[i];
}
media = media / n; printf(“Media = %f\n”, media);
for (i = 0; i<n; i++)
  if (notas[i]>= media)
     printf(“Nota %d: %f\n”, i + 1, notas[i]);
}
```

Calculando o Produto Escalar

Exemplo 3 –Leia dois vetores inteiros de no máximo5 elementos e realize o produto escalar deste vetores.

Sejam v e w dois vetores. O produto escalar <v,w> é o número real (ou inteiro, depende do tipo de dados do vetor) obtido pela soma das multiplicações de cada elemento i dos vetores dados, respectivamente.
Exemplo:
v= [2, 3,-7,5] e
w=[-1,-4,6,-2] então <v,w>=2(-1)+ 3(-4)+(-7)6+5(-2)
=-2-12-42-10=-66

Calculando o Produto Escalar

```
double vetor1[5], vetor2[5], resultado;
int i;
for (i = 0; i < 5; i++) {
   printf("Entre com valor %d para vetor 1: ", i + 1);
   scanf("%lf", &vetor1[i]);
}
for (i = 0; i < 5; i++) {
   printf("Entre com valor %d para vetor 2: ", i + 1);
   scanf("%lf", &vetor2[i]);
}
resultado = 0.0;
for (i = 0; i < 5; i++)
   resultado = resultado + (vetor1[i] * vetor2[i]);
printf("Produto interno: %f\n", resultado);
```

Polinômios

Exemplo 4. Leia os coeficientes e imprima um polinômio de grau n (máximo igual a 10).

```
float coef[11];
int grau, i;
printf("Grau do polinomio (grau maximo = 10): ", &grau);
scanf("%d", &grau);  /*Valide o grau */
for (i = grau; i >= 0; i--) {
    printf("coeficiente de x^%d: ", i);
    scanf("%f", &coef[i]);
 }
printf("%.1fx^%d", coef[grau], grau);
for (i = grau - 1; i >= 0; i--) {
   if (coef[i] != 0)
     if (coef[i] >= 0)
         printf(" + %.1fx^%d", coef[i], i);
     else
         printf(" - %.1fx^%d", -coef[i], i);
 }
```

Polinômios

Exemplo 5. Leia os coeficientes e imprima um polinômio de grau n (máximo igual a 10) e suas derivadas.

```
 float coef[11];
 int grau, i;
/* Le e valida o grau e le os grau+1 coeficientes */
do {
   printf("%.1fx^%d", coef[grau], grau);
   for (i = grau - 1; i >= 0; i--) {
      if (coef[i] != 0)
         if (coef[i] >= 0)
            printf(" + %.1fx^%d", coef[i], i);
         else
            printf(" - %.1fx^%d", -coef[i], i);
   }
   printf("\n");
  for (i = 1; i <= grau; i++)
    coef[i-1] = coef[i]* i; grau--;
} while (grau > 0);
printf("%.1fx^0\n", coef[0]);
```

Busca Básica em um Vetor

Exemplo 6. Ler um vetor v de 10 elementos inteiros e um número num, faça um algoritmo que procure o número num em v e retorne a sua posição, caso ele for encontrado.

Este problema é muito importante em computação pois possui múltiplas aplicações no nosso dia a dia. Por exemplo, localizar um livro na biblioteca, localizar um nome em um cadastro de empregados de uma empresa, saber se um dado CPF está cadastrado em alguma lista de cheques, etc.

Busca Básica em um Vetor

Exemplo 6 (Pseudocódigo)

```
Algoritmo <busca_em_vetor>
inteiro:v[10], x,i, busca
inicio
  para (i ←1; i<=10;i ←i+1)
          leia (v[i])
  fim para
  leia (x)
  busca ← 0
  para (i ←1; i<=10;i ←i+1)
     se (v[ i ] = x) então
          busca←i  /* se for encontrado busca recebe o índice*/
     fim se
  fim para
fim
```

Busca básica em um vetor

O algoritmo faz comparações demais, pois mesmo quando o elemento já foi encontrado o vetor e percorrido ate o final. Podemos modificar o algoritmo para que ele termine sua execução tão logo o elemento seja encontrado (se for encontrado).

```
inteiro:v[10], x,i,busca
logico: achou
inicio
para (i ←1; i<=10;i ←i+1)
    leia (v[i])
fim para
Leia (x)
achou ← 0
i← 1;
```

```
enquanto (i <= 10 e não achou) faça
      se (v[ i ] = x) então
          achou←1
      fim se
      i ←i + 1
fim enquanto
se (achou) então
   busca←i-1
Fim
```

Busca binária em um vetor

Seria possível melhorar se o vetor estivesse ordenado?

Similar a procura de uma palavra no dicionário: Abre no meio:

-Se estiver antes, procura no 1ª metade

-se estiver depois procura na 2ª metade

```
inteiro:v[10], meio, inicio←1, fim←10,x
inicio /*Leia o vetor e o x*/
meio← div(inicio + fim,2) /* meio do vetor */
/*termina quando achou ou quando o fim ficou antes do inicio */
enquanto (v[meio] <> x e inicio<=fim  ) faça
   se (v[meio] > x) então /*joga uma das duas metades fora*/
      fim←meio-1   /* metade superior foi jogada fora */
   senão
      inicio← meio + 1/* metade inferior foi jogada fora */
   fim se
   meio← div(inicio + fim,2) /* recalcula meio */
Fim enquanto
```

```
Se (v[meio] = x) então
   escreva (“Achou
em”,meio)
senão
  escreva (“Não achou”)
Fim se
Fim
```

Vetor de caracteres (string)

- Diferença entre caracteres individuais (char) e texto (string).
- Caracteres individuais:

- Representam apenas um símbolo, letra ou dígito
- Usamos entre aspas simples, exemplos: ‘B', ‘b', ‘z', '4', '.'

- Texto:

- Sequência de caracteres, exemplo: "Algoritmos e Estruturas de Dados“
- Usamos entre aspas duplas.

Strings

- Uma string e sempre terminada pelo caractere especial ‘\0‘. Portanto sempre declaramos uma string com um caractere a mais do que precisa. Exemplo: Se estivermos trabalhando com uma strings de 10 caracteres, deveremos declarar char st[11];
- Exemplo:

A	l	g	o	r	i	t	m	o	s	\0
0	1	2	3	4	5	6	7	8	9	10

Strings – Declaração

- char variavel [tamanho];
  Exemplo: char st[14];

?	?	?	?	?	?	?	?	?	?	?	?	?	?
0	1	2	3	4	5	6	7	8	9	10	11	12	13

- char variavel [tamanho] = "texto";
  Exemplo: char st[14] = “Algoritmos";

A	l	g	o	r	i	t	m	o	s	\o	?	?	?
0	1	2	3	4	5	6	7	8	9	10	11	12	13

- char variavel [] = "texto";
  Exemplo: char st[] = “Algoritmos“;

A	l	g	o	r	i	t	m	o	s	\o
0	1	2	3	4	5	6	7	8	9	10

Strings

- Impressão:

  printf(“%s \n”, st);

- Leitura:

  scanf(“%s”, st)

  – Não tem “&” e não considera brancos e tabs.

  - Para ler strings incluindo espaços usamos: %[^ \n].

  scanf("%[^\n]",st);

- Acesso: por elementos individuais cad[i]

Strings

Exemplo:

```
int main(){
char st[80], st2[80];
int a;
printf("\nEntre com nome sem espaços:");
scanf("%s",st);
printf("\nEntre com nome com espaços:");
scanf("%[^\n]",st2);
printf("\nEntre com idade:");
scanf("%d",&a);
printf("\n Digitado: %s, %s e %d\n",st, st2 ,a);
}
```

Strings

1. Ler uma string de ate 80 caracteres, salvar a inversa desta em um vetor e imprimir a inversa da string lida.

```
int main(){
char st[80], stInv[80];
int tam, i, j;
        printf("Entre com o string: ");
        scanf("%s",st);
        tam = 0;
        while(st[tam] != '\0' && tam < 80){
            tam++;
        }
```

Strings

```
stInv[tam] = '\0';
j = tam-1;
i = 0;
while(i<tam){
        stInv[j] = st[i];
        i++;
        j--;
}
printf("A inversa e: %s\n",stInv);
}
```

Strings

2. Compare duas strings st1 e st2 e imprima se as duas são ou não iguais.

```
#include <stdio.h>
int main(){
        char st1[20], st2[20];
        int i=0;
        printf("Informe duas strings: ");
        scanf("%s %s",st1,st2);
        while(st1[i]==st2[i] && st1[i]!= '\0' && st2[i]!= '\0')
            i++;
        if (st1[i]== '\0' && st2[i]== '\0')
            printf ("As strings sao iguais");
        else
             printf ("As strings nao sao iguais");
}
```

Strings – A biblioteca string.h

Nos exemplos 1 e 2 notamos que para obter o tamanho da string e para comparar duas strings temos que realizar uma programação de baixo nível, pois a manipulação de uma string requer um acesso aos elementos de um vetor de caracteres.

Para poupar tempo e código podemos fazer uso da biblioteca string.h que já possui algumas funções de manipulação de strings sem precisar fazer o acesso a baixo nível.

Strings – A biblioteca string.h

A biblioteca string.h possui funções para

- Descobrir o tamanho de uma string (strlen())
- Comparar strings usando strcmp();
- Copiar strings (strcpy e strncpy)
- Concatenar strings (strcat e strncat)

Strings – A biblioteca string.h

- strlen

Determina o tamanho de uma string

Sintaxe:

variável do tipo inteiro = strlen(string);

- strcmp

Compara o conteúdo de duas strings;

Possíveis valores de retorno:

0: conteúdo das strings são iguais

< 0: conteúdo da string1 é menor do que string2

> 0: conteúdo da string1 é maior do que string2

Sintaxe:

variável do tipo inteiro = strcmp(string1, string2);

Strings – A biblioteca string.h

- strcpy

Realiza a cópia do conteúdo de uma variável a outra.

Sintaxe:

strcpy(string_destino, string_origem)

- strncpy

Realiza a cópia do conteúdo de uma variável a outra, porém, deve ser especificado o tamanho a ser copiado.

Sintaxe:

strncpy(string_destino, string_origem, tamanho)

Strings – A biblioteca string.h

- strcat

Realiza a concatenação do conteúdo de uma variável a outra. Ambas devem ser strings.

Sintaxe

strcat(string_destino, string_origem);

- strncat

Realiza a concatenação do conteúdo de uma variável a outra, porém, deve ser especificado o tamanho a ser concatenado. Ambas devem ser strings.

Sintaxe:

strncat(string_destino, string_origem, tamanho);

Strings – A biblioteca string.h

Ler uma string de ate 80 caracteres, salvar a inversa desta em um vetor e imprimir a inversa da string lida (exemplo 1) com uso da biblioteca string.h

```
#include <stdio.h>
#include <string.h>
int main() {
        char st[80], stInv[80];
        int i, tam;
        printf("Digite uma string: ");
        scanf("%s", st);
        tam = strlen(st);
        printf("A string original eh: %s \n", st);
        for (i = 0; i < tam; i++)
            stInv[tam-1-i] = st[i];
        printf("A string invertida eh: %s \n", stInv);
}
```