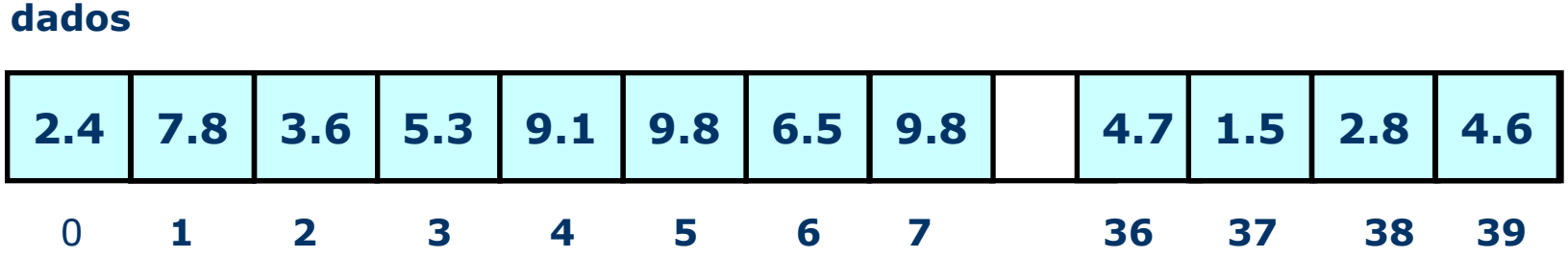
PCI- Vetores

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

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

```
#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));
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]);
}
```

Soma de dois Vetores

Exemplo 3 –Leia dois vetores inteiros de no máximo5 elementos e realize a soma destes vetores.

Sejam v e w dois vetores, a sua soma é outro vetor, formado pelos elementos s[i] que são a soma dos elementos v[i] e w[i]. Exemplo:

v= [2, 3,-7,5] e

w=[-1,-4,6,-2] então

S=[1, -1,-1,3]

Soma de dois Vetores

```
int vetor1[5], vetor2[5], soma[5],i;
for (i = 0; i < 5; i++) {
   printf("Entre com valor %d para vetor 1: ", i + 1);
   scanf("%d", &vetor1[i]);
}
for (i = 0; i < 5; i++) {
   printf("Entre com valor %d para vetor 2: ", i + 1);
   scanf("%d", &vetor2[i]);
}
for (i = 0; i < 5; i++){
   s[i] = vetor1[i] + vetor2[i];
   printf(“s[%d]: %d\n", i, s[i]);
}
```

Produto Escalar

Exemplo 4 –Leia dois vetores de valores reais 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

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

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

Polinômios

Exemplo 6. 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 7. Faça um programa que leia um vetor v de 10 elementos inteiros e um número que deverá ser procurado em v. O programa deverá retornar 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

```
int v[10], x,i, busca= -1;
//Leitura de dados
printf ("Informe os elementos do vetor:\n");
for (i=0; i<10; i++){
   printf ("v[%d] = ",i);
   scanf ("%d", &v[i]);
}
printf ("Informe o numero a ser procurado no vetor: ");
scanf ("%d",&x);
// Busca atraves da comparacao
for (i=0; i<10; i++){
   if (v[i]==x)
      busca=i;
}
// Verifica se a busca deu certo
if (busca>=0)
   printf ("O numero procurado foi achado na posicao %d", busca);
else
   printf ("O numero procurado nao foi achado");
}
```

Busca básica em um vetor

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

Busca básica em um vetor

```
int v[10], x,i, achou=0;
printf ("Informe os elementos do vetor:\n");
   for (i=0; i<10; i++){
      printf ("v[%d] = ",i);
      scanf ("%d", &v[i]);
   }
   printf ("Informe o numero a ser procurado no vetor: ");
   scanf ("%d",&x);
   for (i=0; i<10 && !achou; i++){
      if (v[i]==x)
         achou=1;
   }
if (achou)
      printf ("O numero procurado foi achado na posic %d", i-1);
   else
      printf ("O numero procurado nao foi achado");
}
```

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 na 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
```

Busca binária em um vetor

```
int v[10], i, meio, inicio=0, fim, x,n;
do{
      printf ("Informe o tamanho do vetor [2-10]:\n");
      scanf ("%d",&n);
   }while (n<2 || n>10);
   fim=n-1;
   printf ("Informe os elementos do vetor na ordem crescente:\n");
   for (i=0; i<n; i++){
      printf ("v[%d]=", i);
      scanf ("%d",&v[i]);
   }
   printf ("Informe o elemento a ser procurado:\n");
   scanf ("%d",&x);
   meio=(inicio + fim)/2; /* meio do vetor */
   printf ("Inicio= %d\t Fim=%d\t Meio=%d\n",inicio, fim, meio);
```

```
while (v[meio] !=x && inicio<=fim  ){
      if (x < v[meio])
         fim=meio-1;
      else
         inicio=meio+1;
      meio=(inicio + fim)/2;
}
if (v[meio] == x)
      printf ("Achou na posicao %d",meio);
 else
      printf ("O elemento x nao foi achado no vetor");
}
```

Os exemplos 8, 9, 10 e 11 são algoritmos de ordenação (Algoritmos de Seleção, Bolha e Inserção).

Exemplo 8– Faça um programa que leia um vetor de 20 inteiros e o coloque em ordem crescente, utilizando a seguinte estratégia de ordenação:

•Selecione o elemento do vetor de 20 posições que apresente o menor valor.

•Troque este elemento pelo primeiro.

•Repita estas operações, envolvendo agora apenas os 19 elementos restantes (trocando o de menor valor com a segunda posição), depois os 18 elementos (trocando o de menor valor com a terceira posição), depois os 17,16 e assim por diante, até restar um único elemento, o maior deles. Este algoritmo é conhecido como algoritmo de seleção.

```
Algoritmo <ordemcrescente>
        inteiro:i,j,vetor[20]
        inicio
        para i de 1 até 19 repita
                        menor <-  vetor[i]
                        indice<-  i
                        para j de i+1 até 20 repita
                                se (vetor[j] <  menor ) então
                                        menor<- vetor[j]
                                        indice <-  j
                                fim se
                        fim para
                        se (i<>indice) então
                                vetor[indice] <- vetor[i]
                                vetor[i] <-menor
                        fim se
                fim para
        fim
```

Esse é o algoritmo, faça o programa.

Exemplo 9– Versão um pouco diferente do algoritmo 8 e considerando n elementos (n<=20).

```
Algoritmo <ordemcrescente>
        inteiro:i,j,vetor[20],aux,indice,n
        inicio
                Leia (n)
                para i de 1 até n-1 repita
                        indice <-  i
                        para j de i+1 até n repita
                                se (vetor[j] <  vetor[indice]) então
                                        indice <-  j
                                fim se
                        fim para
                       se (i<>indice) então
                            aux <-vetor[i]
                            vetor[i] <-vetor[indice]
                            vetor[indice]<-aux
                        fim se
                fim para
        fim
```

Implementação em C do algoritmo 9

```
int i,j,vetor[20],aux,n,indice;
  do{
    printf ("Informe o numero de elementos do vetor:");
    scanf ("%d", &n);
  }while (n<1 || n>20);
  for (i=0; i<n; i++){
     printf ("v[%d] = ",i);
     scanf ("%d", &vetor[i]);
  }
  for (i=0; i<n; i++){
          indice=i;
          for (j=i+1; j<n;j++)
                  if (vetor[j] <  vetor[indice] )
                          indice =j;
                  if (indice!=i) {
                          aux=vetor[indice];
                          vetor[indice]=vetor[i];
                          vetor[i]=aux;
                  }
   }
   for (i=0; i<n; i++)
       printf ("v[%d] = %d  ",i,vetor[i]);
```

Exemplo 10– Desenvolva um algoritmo que leia um vetor de n posições inteiras (n<=20) e o coloque em ordem crescente, utilizando como estratégia de ordenação a comparação de pares de elementos adjacentes, permutando-os quando estiverem fora de ordem até que todos estejam ordenados (Algoritmo da Bolha).

(Animação em

http://www.youtube.com/watch?feature=player_embedded&v=gWkvvsJHbwY)

```
Algoritmo <ordemcrescente2>
        inteiro:i,j,aux,vetor[20]
        inicio
                Leia (n)
                para i de 1 até n-1 repita
                        para j de 1 até n-i  repita
                                se (vetor[j] >  vetor[j+1] ) então
                                        aux <-vetor[j]
                                        vetor[j]<-vetor[j+1]
                                        vetor[j+1] <-aux
                                fim se
                        fim para
                fim para
        fim
```

Implementação em C do algoritmo 10

```
int i,j,vetor[20],aux,n;
   do{
     printf ("Informe o numero de elementos do vetor:");
     scanf ("%d", &n);
   }while (n<1 || n>20);

   for (i=0; i<n; i++){
       printf ("v[%d] = ",i);
       scanf ("%d", &vetor[i]);
   }
   for (i=1; i<n; i++)
            for (j=0; j<n-i; j++)
                     if (vetor[j] >  vetor[j+1]){
                              aux = vetor[j];
                              vetor[j]=vetor[j+1];
                              vetor[j+1]=aux;
             }
   for (i=0; i<n; i++)
       printf ("v[%d] = %d  ",i,vetor[i]);
```

Exemplo 11– Desenvolva um algoritmo que leia um vetor de n posições inteiras (n<=20) e o coloque em ordem crescente, utilizando como estratégia de ordenação inserir um elemento k num vetor já ordenado de k-1 elementos.

```
Algoritmo <ordemcrescente3>
        inteiro:i,j,elemento,vetor[20]
        inicio
                Leia (n)
                para j de 2 até n repita
                        elemento <-vetor[j]
                        i <-j-1
                        enquanto (i>0 e vetor[i]>elemento)
                                vetor[i+1] <- vetor[i]
                                i <-i-1
                        fim enquanto
                        vetor[i+1] <-elemento
                fim para
        fim
```

Implementação em C do algoritmo 11

```
int i,j,vetor[20],elemento,n;
  do{
    printf ("Informe o numero de elementos do vetor:");
    scanf ("%d", &n);
  }while (n<1 || n>20);
  for (i=0; i<n; i++){
      printf ("v[%d] = ",i);
      scanf ("%d", &vetor[i]);
  }
  for (j=1; j<n; j++){
          elemento=vetor[j];
          i=j-1;
          while (i>=0 && vetor[i]>elemento){
                    vetor[i+1]=vetor[i];
                    i--;
          }
          vetor[i+1]=elemento;
   }
  for (i=0; i<n; i++)
         printf ("v[%d] = %d  ",i,vetor[i]);
```

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 é 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:

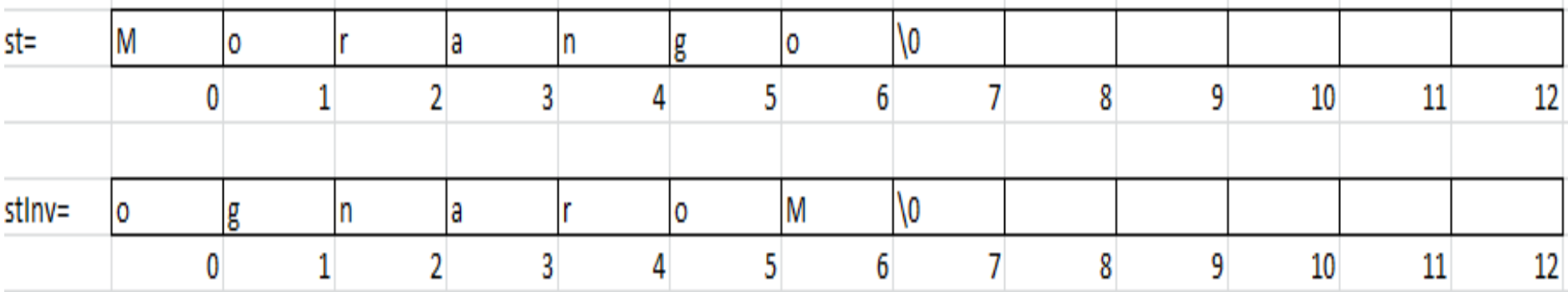
```
int main(){
    char st[80], st2[80];
    int a;
    printf("\nEntre com nome sem espaços:");
    scanf("%s",st);
    printf("\nEntre com nome com espaços:");
    scanf("%[^\n]",st2);
    printf("\nEntre com idade:");
    scanf("%d",&a);
    printf("\n Digitado: %s, %s e %d\n",st, st2 ,a);
}
```

Observe que ao ler o nome sem espaços o caractere final enter será armazenado como conteúdo de st2. Evitamos esse problema escrevendo “ %[^\n]” (adiciona espaço em branco entre “ e %) ou colocando fflush(stdin); entre um scanf e outro

Strings

1. Ler uma string de até 13 caracteres, determinar e imprimir a inversa da string lida.

```
int main(){
    char st[13], stInv[13];
    int tam, i, j;
    printf("Entre com o string: ");
    scanf("%s",st);
    tam = 0;
    while(st[tam] != '\0' && tam < 13){
            tam++;
   }
   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);
}
```