Linguagem de Programação II

Prof. Mario Bessa

Aula 9

http://mariobessa.info

Arrays

- **Motivação:**
  - O programa abaixo calcula a <u>média</u> de N notas.

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

```
float nota,media,soma=0.0;
int i,N;

printf("Entre com a quantidade: ");
scanf("%d",&N);
for(i=1; i<=N; i++){
    scanf("%f",&nota);
    soma += nota; // soma = soma + nota;
}
media = soma/N;
printf("Media: %.2f\n",media);
```

Arrays

- **Motivação:**
  - E para calcular o desvio padrão?

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

  - Para usar a equação acima precisamos primeiro calcular a média e depois percorrer novamente cada nota a fim de calcular o somatório acima.
  - Como fazer?
    - Pedir para o usuário digitar novamente os dados.
      - **Extremamente deselegante.**
    - Armazenar cada nota em uma variável.
      - **Inviável.**

Arrays

- **Motivação:**
  - Calcular o desvio padrão
    - Armazenar cada nota em uma variável (inviável).

```
float nota,media,soma=0.0;
float nota1,nota2,nota3, ...
int i,N;

scanf("%d",&N);
for(i=1; i<=N; i++){
   scanf("%f",&nota);
   soma += nota;
   if(i==1)       nota1 = nota;
   else if(i==2) nota2 = nota;
   else if(i==3) nota3 = nota;
   ...
}
media = soma/N;
...
```

Arrays

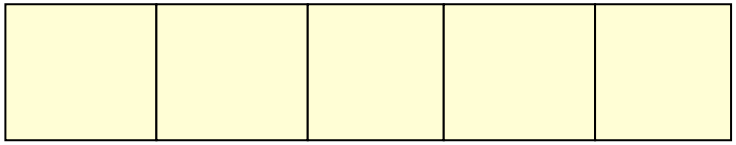
- **Definição:**
  - Um Tipo Estruturado Homogêneo é um tipo composto por vários elementos de mesmo tipo básico.
  - Array é uma série de variáveis do mesmo tipo referenciadas por um único nome, onde cada variável (elemento) é diferenciada através de um número chamado "índice".
  - Os elementos do array são guardados em posições consecutivas de memória (alocação sequencial).

Arrays

- **Declaração:**
  - Array unidimensional (**vetor**)

    Sintaxe: <tipo> <nome> [tamanho];

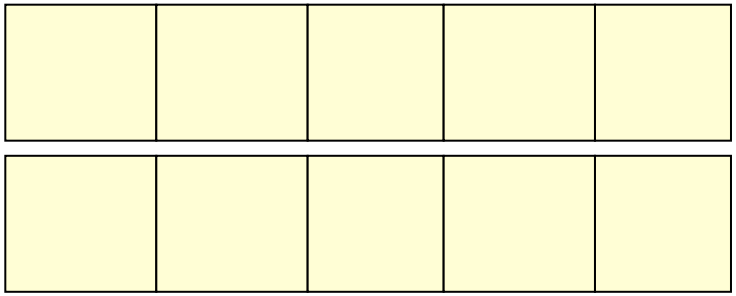
    Ex: float notas[5];



  - Array bidimensional (**matriz**)

    Sintaxe: <tipo> <nome> [tamanho][tamanho];

    Ex: float salario[2][5];



  - Array multidimensional

    Sintaxe: <tipo> <nome> [tamanho][tamanho]...[tamanho];

    Ex: float valor[2][4][3];

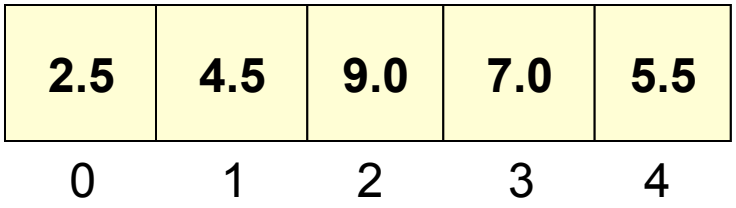
Vetores

- **Definição:**

```
int main(){
   float notas[5];

   notas[0] = 2.5;
   notas[1] = 4.5;
   notas[2] = 9.0;
   notas[3] = 7.0;
   notas[4] = 5.5;
   return 0;
}
```

Representação gráfica:



Vetores

- **Definição:**

```
int main(){
   float notas[2][4];

   notas[0][0] = 2.5;
   notas[0][1] = 4.5;
   notas[0][2] = 9.0;
   notas[0][3] = 7.0;
   notas[1][1] = 5.5;
   notas[1][1] = 8.5;
   notas[1][1] = 5.5;
   notas[1][1] = 6.5;
   return 0;
}
```

Representação gráfica:

0	**2.5**	**4.5**	**9.0**	**7.0**
1	**5.5**	**8.5**	**5.5**	**6.5**
	0	1	2	3

Vetores

Declaração:

```
tipo identificador[tamanho];
```

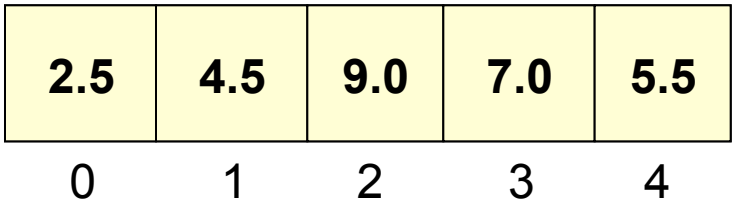
- **tamanho** deve ser uma **constante inteira**!

```
int N=5;
float notas[N];
```

```
int main(){
  float notas[5];

  notas[0] = 2.5;
  notas[1] = 4.5;
  notas[2] = 9.0;
  notas[3] = 7.0;
  notas[4] = 5.5;
  return 0;
}
```

Representação gráfica:



Vetores

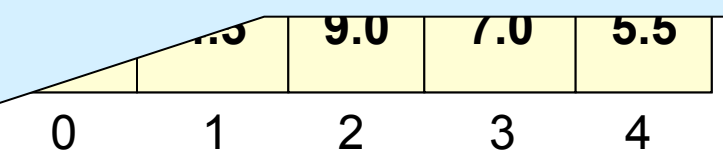
```
int main(){
  float notas[5];

  notas[0] = 2.5;
  notas[1] = 4.5;
  notas[2] = 9.0;
  notas[3] = 7.0;
  notas[4] = 5.5;
  return 0;
}
```

Inicializando um elemento:

`identificador[índice]`

Índices **válidos** entre **0** e **tamanho-1**.
Cuidado: C não realiza verificação de limites. Acessos errados acarretam resultados imprevisíveis.



Vetores

- **Definição:**
  - O primeiro elemento em cada vetor é identificado pela posição elemento zero.
  - Assim, o primeiro elemento do vetor x é referenciado como x[0], o segundo elemento, como x[1], o sétimo elemento x[6], e, em geral, o i-ésimo elemento do vetor x é referenciado como x[i – 1].
  - Os nomes de vetores, como outros nomes de variáveis, só podem conter letras, dígitos e sublinhados.

Vetores

- **Definição:**
  - Se um programa usa uma expressão como um subscrito, então a expressão é avaliada para determinar o subscrito.
  - Por exemplo, se a = 5 e b = 6, então a instrução

```
c[ a + b ] += 2; // soma 2 ao elemento do array c[11]
```

Inicialização de Vetores

– Elemento a elemento:

```
double balance[5];
balance[0] = 1000.0;
balance[1] = 2.0;
balance[2] = 3.4;
balance[3] = 17.0;
balance[4] = 50.0;
```

```
int mat [2] [3];
mat[0][0]=1;
mat[0][1]=2;
mat[0][2]=3;
mat[1][0]=4;
mat[1][1]=5;
mat[1][2]=6;
```

– Ou na declaração:

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
int mat [2] [3] = {1, 2, 3, 4, 5, 6};
int mat [2] [3] = { 1, 2, 3};
```

Vetores

- **Inicializando vetores:**
  - O número de valores entre { } não pode ser maior do que o número de elementos declarados entre [ ].
  - Se for omitido o tamanho do vetor, um vetor com um tamanho suficiente é criado na inicialização:

    double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};

    balance terá o tamanho de 5

  - Se houver menos inicializadores que elementos no vetor, os elementos restantes serão inicializados em zero.
  - É importante lembrar que os vetores não são automaticamente inicializados em zero.
  - Você precisa, pelo menos, inicializar o primeiro elemento em zero para que os elementos restantes sejam automaticamente zerado.

Vetores

- **Definição:**
  - C não faz a verificação dos limites do vetor para impedir que o programa se refira a um elemento que não existe.
  - Assim, um programa em execução pode ultrapassar o final de um vetor sem aviso.
  - Você deverá garantir que todas as referências de um vetor permaneçam dentro dos seus limites.

Vetores

- **Problema:**
  - Calcular o desvio padrão das notas digitadas.

```
#define LIM 500
int main(){
  float media,desvio,soma=0.0,notas[LIM];
  int i,N;
  scanf("%d",&N);
  for(i=0; i<N; i++){
    scanf("%f",&notas[i]);
    soma += notas[i];
  }
  media = soma/N;
  soma = 0.0;
  for(i=0; i<N; i++)
    soma += (notas[i]-media)*(notas[i]-media);
  desvio = sqrtf(soma/N);
  printf("Desvio: %.2f\n",desvio);
  return 0;
}
```

Vetores

- **Problema:**
  - Calcular o desvio padrão das notas digitadas.

```
#define LIM 500
int main(){
  float media,desvio,som
  int i,N;
  scanf("%d",&N);
  for(i=0; i<N; i++){
    scanf("%f",&notas[i]);
    soma += notas[i];
  }
  media = soma/N;
  soma = 0.0;
  for(i=0; i<N; i++)
    soma += (notas[i]-media)*(notas[i]-media);
  desvio = sqrtf(soma/N);
  printf("Desvio: %.2f\n",desvio);
  return 0;
}
```

Índice pode ser especificado por uma **expressão inteira** ou pelo conteúdo de uma **variável inteira**.

Vetores

- **Problema:**
  - Calcular o desvio padrão das notas digitadas.

```
#define LIM 500
int main(){
  float media,desvio,so
  int i,N;
  scanf("%d",&N);
  for(i=0; i<N; i++){
    scanf("%f",&notas[i
    soma += notas[i];
  }
  media = soma/N;
  soma = 0.0;
  for(i=0; i<N; i++)
    soma += (notas[i]-media)*(notas[i]-media);
  desvio = sqrtf(soma/N);
  printf("Desvio: %.2f\n",desvio);
  return 0;
}
```

```
if(N>LIM){
  printf("Tamanho excedido\n");
  //Encerra o programa.
  return 0;
}
```

Vetores

- **Definição:**
  - Em **C**, operadores de atribuição (como **=** e **+=**) não são aplicáveis à vetores.
  - Para se atribuir (copiar) o conteúdo de um vetor **a** para o vetor **b**, por exemplo, deve-se fazer um laço (**loop**) e atribuir elemento a elemento:

```
for (i=0; i<10; i++)
    b[i] = a[i];
```

Vetores

- **Definição:**
  - A Linguagem C provê funções especiais para manipulação de vetores (na verdade, manipulação de memória). Para isso, deve-se acrescentar **#include <memory.h>** .
  - A função **memmove(b, a, sizeof(a))** copiará todo o conteúdo do vetor **a** para o vetor **b**.
    - **sizeof(x)** retorna o tamanho em bytes ocupado por x.
  - A função **memset(a, 0, sizeof(a))** atribuirá o valor 0 a todos os elementos do vetor **a**.

Vetores

```
#include <stdio.h>
#include <memory.h>
int main(){
    int v1[]={1,2,3,4,5},v2[5];
    memmove(v2, v1, sizeof(v1));
    for (int i=0; i<5; i++) {
        printf("\nv2[%i]=%i",i,v2[i]);
    }
    return 0;
}
```

Vetores

```
#include <stdio.h>
#include <memory.h>
int main(){
    int v1[5];
    for (int i=0; i<5; i++) {
        printf("\nv1[%i]=%i",i,v1[i]);
    }
    memset(v1, 0, sizeof(v1));
    for (int i=0; i<5; i++) {
        printf("\nv1[%i]=%i",i,v1[i]);
    }
    return 0;
}
```

Vetores

- **Problema:**
  - Escreva um programa em C que leia um vetor com 10 posições de números inteiros e verifique se um determinado valor, também digitado pelo usuário, está no vetor.

```
#include <stdio.h>
#define tam 10
int main() {
   int vetor[tam], valor, i, achou=0;
   for (i=0;i<tam;i++)
      scanf("%d", &vetor[i]);
   scanf("%d", &valor);
   for (i=0;i<tam;i++){
      if (valor==vetor[i])
         achou=1;
   }
   if (achou)
      printf("O valor %d está no vetor.", valor);
   else
      printf("O valor %d não está no vetor.", valor);
   return 0;
}
```

Vetores

- **Problema:** .
  - Reescreva o programa anterior e informe o índice do número lido no vetor, caso exista. Caso o elemento não esteja no vetor, apresente uma mensagem informando tal situação.

```
#include <stdio.h>
int main(){
    int i,v1[5],n,achou=0;
    for (i=0; i<5; i++) {
        printf("\nEntre com um valor ");
        scanf("%i",&v1[i]);
    }
    printf("\nEntre com o valor a ser pesquisado ");
    scanf("%i",&n);
    for (i=0; i<5; i++) {
        if (v1[i]==n) {
            achou=1;
            break;
        }
    }
    if (achou)
        printf("O valor está no vetor na posição %i",i);
    else
        printf("O valor não está no vetor");
    return 0;
}
```

Vetores

- **Problema:**
  - Escreva um programa em C que leia dois vetores de 10 posições e faça a multiplicação dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Mostre o vetor resultante.

```
#include <stdio.h>
int main(){
    int v1[10],v2[10],v3[10];
    printf("Leitura do primeiro vetor\n");
    for (int i=0; i<10; i++) {
        printf("Entre com um valor ");
        scanf("%i",&v1[i]);
    }
    printf("Leitura do segundo vetor\n");
    for (int i=0; i<10; i++) {
        printf("Entre com um valor ");
        scanf("%i",&v2[i]);
    }
// multiplicação dos vetores
    for (int i=0; i<10; i++)
        v3[i]=v1[1]*v2[i];
// impressão do tercero vetor
    printf("Terceiro vetor\n");
    for (int i=0; i<10; i++) {
        printf("v3[%i]=%i\n",i,v3[i]);
    }
    return 0;
}
```

Vetores

- **Problema:**
  - Escreva um programa em C que leia um vetor de 20 posições e mostre-o. Em seguida, troque o primeiro elemento com o último, o segundo com o penúltimo, o terceiro com o antepenúltimo e assim sucessivamente. Mostre o novo vetor depois da troca.

```c
#include <stdio.h>
#define LIM 6
int main(){
    int v1[LIM],aux;
    printf("Leitura do vetor\n");
    for (int i=0; i<LIM; i++) {
        printf("Entre com um valor ");
        scanf("%i",&v1[i]);
    }
    printf("Impressão do vetor\n");
    for (int i=0; i<LIM; i++) {
        printf("%i",v1[i]);
    }
    // troca do vetor
    for (int i=0; i<LIM/2; i++){
        aux=v1[i];
        v1[i]=v1[LIM-1-i];
        v1[LIM-1-i]=aux;
    }
    // impressão do vetor
    printf("\nImpressão do vetor alterado\n");
    for (int i=0; i<LIM; i++) {
        printf("%i",v1[i]);
    }
    return 0;
}
```

Vetores

- **Problema:**
  - Faça um programa para preencher 2 vetores de 20 posições de inteiros cada, sendo um vetor SOMENTE com números ímpares (VI) e outro SOMENTE com números pares (VP).
  - Em seguida, gerar um terceiro vetor (VR) cujos elementos sejam os elementos de VI e VP intercalados, iniciando com o primeiro elemento de VI.
  - No final imprimir o vetor resultante (VR), o vetor de impares(VI) e o vetor de pares(VP).

    OBS: o programa deverá ser capaz de ler os elementos e a partir da leitura fazer a separação de pares e ímpares. O programa deverá garantir que tanto VP quanto VI sejam totalmente preenchido (sem posições em branco)
  - O vetor VR deve ser preenchido em um único laço, evitando-se um laço para preenchimento primeiro com os elementos ímpares(VI) e depois outro laço para preenchimento dos elementos pares (VP).
  - Exemplo de um vetor de 5 posições:
    - VI = 1 3 5 7 9
    - VP = 2 4 6 8 10
    - VR = 1 2 3 4 5 6 7 8 9 10

```
#include<stdio.h>
main(){
    int vp[5],vi[5],vr[10];
    int n,i=0,p=0,ii;
    printf ("Digite 5 n. pares e impares");
   do{
       scanf("%d",&n);
       if (n%2==0){
           vp[p]=n;
           p++;}
        else{
           vi[i]=n;
           i++;
           }
    }while ((p<5) || (i<5));
    for (i=0,ii=0;ii<10;ii+=2,i++){
       vr[ii]=vi[i];
       vr[ii+1]=vp[i];
     }
printf("\n\n vetor resultante: \n");
for(i=0;i<10;i++)
    printf("%d ",vr[i]);
printf("\n \n vetor de ímpares \n");
for (i=0;i<5;i++)
    printf("%d ",vi[i]);
printf("\n \nvetor de pares \n");
for (i=0;i<5;i++)
    printf("%d ",vp[i]);
}
```

Vetores

- **Problema:** Faça um programa para preencher dois vetores X e Y (com valores lidos do teclado) de 40 posições. O programa deve atribuir a um vetor Z os valores de X e Y intercalados, de forma que a 1ª posição de Z terá um valor de X (primeira posição), a 2ª posição de Z terá um valor de Y (última posição), a 3ª posição de Z terá um valor de X (2ª posição), a 4ª posição de Z terá o penúltimo elemento de Y e assim sucessivamente até preencher o vetor Z.

X	1	3	5	7	10

Y	12	8	6	4	2

Z	1	2	3	4	5	6	7	8	10	12

```
#include <stdio.h>
#define LIM 3
int main(){
    int x[LIM],y[LIM],z[2*LIM];
    int ix=0,iy=LIM-1;
    printf ("\nDigite o primeiro vetor ");
    for (int i=0;i<LIM; i++){
        scanf("%i",&x[i]);
    }
    printf ("\nDigite o segndo vetor ");
    for (int i=0;i<LIM; i++){
        scanf("%i",&y[i]);
    }
    for (int i=0; i<2*LIM; i+=2) {
        z[i]=x[ix];
        z[i+1]=y[iy];
        ix++;
        iy--;
    }
    printf ("\nTerceiro vetor ");
    for (int i=0;i<2*LIM; i++){
        printf("%2i",z[i]);
    }
return 0;
}
```

Vetores

- **Problema:**
  - Escreva um programa em C que leia um vetor de 20 posições. Imprima o maior valor desse vetor e sua posição.

```
#include<stdio.h>
#define LIM 20
main(){
    int V[LIM],maior,posicao;
    printf("Entre com %i valores para o vetor\n",LIM);
    for (int i=0; i<LIM; i++) {
        scanf("%i",&V[i]);
    }
    maior=V[0];
    for (int i=0; i<LIM; i++) {
        if (V[i]>maior) {
            maior=V[i];
            posicao=i;
        }
    }
    printf("Maior valor é %i e está na posição %i",maior,posicao+1);
}
```